C++

Basic

Old computer language

* Machine language - 0, 1
* Assembly language

High level language

* C
* C++
* Python .. etc
* Java
* JS

* linker - A linker is an important utility program that takes the object files, produced by the assembler and compiler and other code to join them into a single executable file. There are two types of linkers, dynamic and linkage

1. What is Compiler ?
   See in Y and Q Chapter

2. What is source code ?
   See in Y and Q Chapter

<iostream> - input, output.

std - standard

rules

```
int main ()
{
  return 0;
}
```

* ; - full stop or end
* library file - It is a ready made file of a programs. It helps us to do a program fast.

If you use a library file your could use : #include < library file >

cout - Character out

out - <<

```
#include<iostream>
  int main ()
{
  std::cout<<"welcome";
  return 0;
}
```

If you need to a character type a word like hello, hi etc.. you need to use character (" ") your word"

Variables

date - read / write / change =
storage as value

eg: whatsapp
a ---- b
Raj --- Vimal == message is date.

It is storage in someplace is called value
It is storage in a variable

a = 5 ;
Variable value

Variable can be create by
a, apple, app121, ass - , etc
- correct
12 raj, @ * & % - incorrect

data type

* int storage = 1 to infinite
* char store = a to z letters,
raju words King sentence hi how are you.

assignment operator (=)

types of assignment

* int a = 5;
* int b;
int b = 5;
* int a, b;
int a = 5, b = 5;

Comment :-
// use this so you can write 1 line comment.
/* start
*/ end between this you can write big paragraph. You compiler donot take as a program.

What is syntax? Y and Q chapter

What is function Y and Q chapter.

what is a variable
A variable is a value that can change, depending on conditions or on information passed to the program.

double quotes " " for character
int donot need ;)
single letter = single quotes ' '

You create a variable a and another a

```
#include <iostream>
int main()
{
    int number = 5;
    std::cout << number;
    return 0;
}
```

```
#include <iostream>
int main()
{
    char number = 'S';
    std::cout << number;
    return 0;
}
```

```
int main() #include <iostream>
{
    int a = 5;
    int b = 10;
    int a1 = 15;
    int b2 = 20;
    std::cout << a+b+a1+b2;
    return 0;
}
```

is ~~ot incor~~ incorrect
You need to create like this
a , a1

(,) used for Separate Variables

```
#include <iostream>
int main()
{
    int n1, n2
    int n1 = 10, n2 = 20;
    int addvalue = n1 + n2;
    int divvalue = n1 / n2;
    int finalvalue1 = addvalue * 5;
    int finalvalue2 = divvalue * 5;
    return
```

(+) It is ~~work~~ work same as Maths

Mathematical operater
+, -, *, /, etc...

```
std::cout << F1
<< "\n";
```

```
std::cout << finalvalue1;
std::cout << finalvalue2;
return 0;
}
```

another Ex

```
#include<iostream>
int main()
{
int n1=10, n2=20
Std::cout<<"answer is:
"<<(n1+n2)*5<<"\n";
return 0;
}
```

```
std::cout<<"answer is:"
<<(n1/n2)*5;
```

"\n" used if a value need go to another line.

donot do like this
int a;

if if you donot give value computer fix a 00 or another number, not error.

```
float store = 1.2, 3.5
```

input ->>

~~Std:cin~~ =

cin - character input

```
#include<iostream>
int main()
{
int n1, n2
Std::cout<<"give ~~answer~~ value for n1 and n2";
std::CIN>>n1>>n2;
```

```
std::cout<<"answer 1 is:"<<(n1+n2)*5
    <<"\n";
std::cout<<"answer 2 is:"<<(n1/n2)*5;
    return 0; }
```

How to create a function

```
int getdata() {
    std::string Name;
    int age;
    std::cout<<"Write your name
    and age";
    std::cin>>Name>>age;
        return 0;
    }
```

```
int main()
{
    std::cout<<"Hi\n";
    getdata();
}
```

donot do like this
If you write function down at int main() compiler will show error ~~and tik also like int main()~~

```
int value() {
}
```

what is Keywords? see 1 and 2 chapter

function - It helps to reuse our program

getdata you can give any name but donot give Keywords. You need to give data type. int

If you use a function in computer ~~we must~~ will understand using () Ex: int main(), getdata().

int getdata() {

} ~~it is called~~ this called function

6. What is Caller? 1 and 2 cha

getdata() this function is called caller

There two type of functions.
One defined. Ex: #include<iostream> as predefined
Ex: int getdata()
{

}

donot do like this
donot write a function
like this

```
int main()
{
    int getdata() {

    }
```

It will also end in error.
correct thing do is write a function up int main() to work correct. but you can also write down int main(). You need write Your function up int main() like this.

int getdata(); is called Function Prototype

```
#include <iostream>
#include <string>
int add(int a, int b);
int main()
{
    std::count << "Hi";
    add(5,2); //this arguments
    return 0;
}
```

8. What is Argument Yand Q

Function rule is at a function a another functi can't be create

7. What is function Prototype? Yand Q cha

but you can call functions.

```
int add (int a, int b) // parameter
{
    std::cout<<a+b<<"\n";
    return 0;
}
```

9. What is Parameter? y and a cha

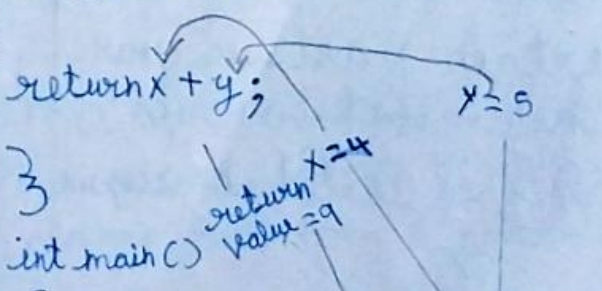
You add How many Parameter You want. int add (int a, int b, int c) but P = A
How many you add you also add in arguments. add(5,7,8)
If you use Function Prototype your need to add same number as your parameter and Arguments.

You can use all Mathematical Operater. Ex

```
std::cout<a+b-c*d/e<<"\n";
```

Map of Parameter and Arguments

```
#include <iostream>
int add (int x, int y)
{
    return x+y;
}
int main()
{
    std::cout<<add(4,5)<<
    std::endl;
    return 0;
}
```



What is Void main()?

The void main() indicates that the main() function will not return any value, but the int main() indicates that the main() can return integer type data. When our program is simple, and it is not going to terminate before reaching the last line of the code, or the code is error free, then we can use the void main().

What is scope?

When you declared a program element such as a class, function or variable, its name can only be "seen" and used in certain parts of your program. The context in which a name is visible is called its scope.

{ this called block }

what is life time ?

Every object and reference has a lifetime, which is a runtime property: For any object or reference, there is a point of execution of a program when its lifetime begins, and there is a moment when it ends.

DRY - Don't repeat Yourself

So there introduction functions ~~makes~~ to do it repeat.

What is Forward Declaration?

Forward Declaration allows You to declare an identifier before it is defined. This means you can inform the compiler about a class, function, or variable before it is used in code. A declaration statement without a definition is used to achieve this.

You can a functions with a single name. You can't use the same name to create the another functions

What is ~~pre~~ preprocessor?

~~When~~ which start with # is called preprocessor

When run a program preprocessor will run first because it contains library files. You use in your Program

Ex: #include <iostream>, #include <string>

What is string?

A string is a Variable that stores a sequence of letters or other characters, such as "Hello" or "May 10th is my birthday. to create a String we ~~first~~ first declare it. #include <string>

#define Name "raj"
NAME Variable Value

If you use define. ~~You Variable~~ Your give Variable will print the give & value only.

When You use Name the raj fixed value will print through your program.

while you use define the variable only in the capital letter

How create your own library files?

* #include "calculater.h"
  What name you want to give is your choice

* You Need to create a text Document
* You Need to Name Calculater.h
* You open with code editor

```
#include <iostream>
add (int a, int b) {
    return a+b;
}
```

* And create a another source file

```
#include<iostream>
#include "calculater.h"
int main() {
    std::cout<<add
    (4,5);
    return 0;
}
```

* Save the file with what name you want.

* And run the program the result is 9.

* You can write for sub, div, Mul and anything.

* You need to keep the tri library file and saved in a same place like folder.

Type	Bits	Range
int	16	-32,147,483,647 to 2,147,483,647
Unsigned int	16	0 to 4294967295
Signed int	16	-2147483648 to 2147483647
short int	16	-32,768 to 32,767
Unsigned short int	16	0 to 65,535
Signed Short int	16	-32,767 to 32,767
long int	32	-2,147,483,647 to 2,147,483,647
Unsigned long int	32	0 to 4,294,967,295
Signed long int	32	-2,147,483,647 to +2,147,483,647
float	32	1.175494351 E-38, 3.402823466 E+38
double	64	2.2250738585072014 E-308, 1.7976931348623158 E+308.
long double	80	3.4E-4932 to 1.1E+4932
char	8	-128 to 127
unsigned char	8	0 to 255
signed char	8	-128 to 127

Unsign can store 0 and +
Signed can store + and - and 0.

Computer can understand two things
one is True - 1
another is False - 0

If True computer take as 1. If False it take as 0.

boolean is a data type. It can store True and False. It is keyword is bool.

Sizeof() is a Keyword, but it is a compile-time operater that determines the size, in bytes, of a variable or data type.

```
int main()
{
  int a = 111;
  Std::cout << sizeof(a);
}
```

result = 4

General File sizes.
bit = 1
byte = 8 bit
Kb = 8000 byte
mb = 1024 Kb
gb = 1024 mb
tb = 1024 tgb

Size of a data types is are fixed.
int - 4 byte
char - 1 byte

1 byte = 8 bit

ASCII (American Standard Code for Information Interchange) is the most common character encoding format for text data in computers and on the Internet. In standard ASCII-encoded data, there are unique values for 128 alphabetic, numeric or special additional characters and control codes.

How to find ASCII code

A = 65
a = 97
B = 66
b = 98
C = 67
c = 99
	#include <iostream>
	int main() {
	char b = 'a';
	std::cout<<(int)b;
	}

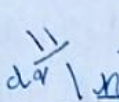
What is escape sequence?

Character combinations of a backslash (\) followed by a letter or by a combination of digits are called "escape sequence". To represent a newline character, single quotation mark, or certain other characters in a character constant,

you must use ~~to~~ escape sequences.

Ex : \n - new line
\t - Horizontal Tale.
etc...

How to use escape sequence



* use Back slash (\)
* use double quotes (" ")

ESS - Escape sequences

```
#include <iostream>
#include <string>
int main()
{
    Std:: string name;
    std::cin >>name;
    std::cout<< name;
}
```

only with cin

disadvantages of cin :

Its is because cin will not check the length of the string, and it breaks after you enter a blank space.

with a Example :

output Tale
My name is raj
My

```
int main()
{
    std::string name;
    std::getline(std::cin, name);
    std::getline cout << name;
}
```

getline + cin

get line() use for to avoid the cin disadvantages. getline()

output Tab

only cin	get line + cin
My name is Raj My	My name is Raj My name is Raj

If you use cin and getline + cin in a one block {, }. It will be error.

It will execution the first cin. or get line + cin. and ignore the get line + cin or cin only.

If we use get line first and use cin second. It will (works). How it works in output tab

output tab
raj raj
raj raj
tom t
tom

What is getline?
getline() is an in-built function defined in the <string> header file that allows accepting and reading single and multiple line strings from the input stream. The cin object also allows input from the user, but not multi-word or multi-line input.

10. What is Header file? y and Q chapter.

11. different between header file and library?

Example Program of ignore

```
int main() {
    std::string name, name1;
    std::cin >> name;
    std::cout << name;
    std::getline(std::cin, name);
    std::cout << name1;
}
```

std::
cin.ignore
();

~~If you use this it will not work correct~~ but you need to write like this in output Yale.

output Yale
ra j
ra j

If ignore name.
output comes is name 1. Sometimes name output comes and ignore name 1.

If you use std::cin.ignore(); cin and getline() works will be correct but....

output Yale
r
r raj
raj

another things can do with string

```
int main() {
    std::string name,
    name1;
    std::cin >> name;
    std::cin >> name1;
    std::cout << name + name1;
}
```

output Yale
sri
sulan
srisulan

```
int main() {
    std::string name;
    std::cin >> name;
    std::cout << name.length();
}
```

output
raj
3

What is String length()?

This function is used to find the length of the string in terms of bytes. This is the actual number of bytes that conform the contents of the string, which is not necessarily equal to the storage capacity.

enum data types is a user defined ~~user~~ data types. To create this enum is Keyword and your give a data (on your own) name.

Example (to create)

```
enum color (give any name, you want)
{
    WHITE,   (rule)
    BLACK,   (rule)
    GREEN
};   (this also a rule)
```

when you create a data type enum. rules

```
{
    every letter could be Capital;
};
```

~~to list~~

color ~~list~~ (or Var) = ~~black~~ BLACK; this also could be capital letter

~~Std::cout << first;~~

User without assigned a value computer will assigned :-

~~int~~ WHITE, = 0

BLACK, = +1

GREEN = +1

If user assigned a value it will be plus with the value. Next is not assigned it will be plus 1. ~~If user give it will be plus with user give value.~~

WHITE = 5,

BLACK, = 5

GREEN = 10

to

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

```
enum color {
    WHITE=5,
    BLACK=0,
    RED
};
int main() {
    color First=WHITE;
    cout<<First;
}
```

```
#include<iostream>
using namespace std;
struct person {
    int id;
};
int main(void) {
    struct person s1;
    s1.id=1;
    cout<<s1.id;
}
```

12. difference between function and struct. see Y and Q

13. What is struct? see Y and Q

14. What is int main(void)? see Y and Q

```
#include <string>
#include <iostream>

struct person {
    int id;
    std::string name;
};

int main(void) {
    struct person S1;
    S1.id = 1;
    S1.name = "raj";
    std::cout << S1.name;
```

```
int main(void) {
    Struct Person S1;
    S1.id = 1;
    S1.name = "raj";
    person S2 = {2, "rose"};
    Std::cout<<S2.id<<"\n";
    std::cout<<S2.name;
}
```

Person S2; is a another way of declaration

What is control flow?

Programming is flow control, which refers to the ability to direct the flow of a program based on specific conditions. This allows developers to control how their programs execute and can help to make them more efficient and effective.

exit (0); is this used to stop execution or running. and this halt statement.

exit (0);

The exit function, declared in <Stdlib.h>, terminates a C program. The value supplied as an argument to exit is ~~return code or exit~~ code returned to the operating

systemas the program's return code or exit code

What is halt?

A halt is a flow control statement that terminates the program. halts are implemented as functions (rather than keywords), so our halt statements will be function calls.

What is a jump statement?

Jump statements allow you to skip exit a loop, start the next iteration of a loop, or explicitly transfer program control to a specified location in your program. There is four Jump statement:

* goto * break
* return * Continue

What is goto statement?

The goto statement is used for altering the normal sequence of program execution by transferring control to some other part of the program.

What is return statement?

Terminates the execution of a function and returns control to the calling function (or to the operating system if you transfer control from the main function). Execution ~~resu~~ resumes in the calling function at the point immediately following the call.

What is break statement?

The break statement ends execution of the nearest enclosing loop or conditional statements in

Which it appears. Control passes to the statement that follows the end of the statement, if any.

What is continue statement?

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next ~~iter~~ iteration in the loop.

What is conditional branching?

A Conditional branch instruction is a branch instruction that may or may not generate a transmission of control that relies upon the value of stored bits in the PSR (processor status register). It provides decision-making capabilities in the control unit.

Conditional branching Key word is if ( ) .

Keyword your conditional

Example

```
int main () {

    int a = 5, b = 4;

    if (a == b)
    std::cout << "a and b are equal";
    }
```

if the conditional is correct they the compiler will print a and b equal

if the conditional is incorrect the compiler will ~~print nothing~~ stop there or exit.

What is (==) ?

The equal-to operator (==) returns True if both operands have the same value; otherwise, it ~~retur~~ returns ~~fa~~ false.

If you want to tell something if it's conditional False. else is a Key word. see in example.

Example

```
int main() {
    int a=5, b=4;
    if (a==b)
    std::cout<<"a and b are
    equal";
    else
    std::cout<<"a and b are
    not equal";
}
```

Why we use %o for find odd or even?

If answer is 2 it even but we give 5.5 is odd. so we use %o

$$2\overline{)5}\ (2,\ 4,\ 1$$

we use / this answer.

we use %o this answer.

What is difference between %o and /?

/ is used for division

% is used for Modulus

How to Find odd or even

```
int main() {
    int a=5, b=4;
    if (a%2==0)
    std::cout<<"a is even";
    else
    std::cout<<"a is odd";
}
```

What is %?

C++ provides the modulus operator, %, that yields the remainder after integer division. The modulus operator can be used only with integer operands. The expression x%y yields the remainder after x is divided by y.

```
Struct vote {
    int age;
};
int main() {
    struct vote vot;
    std::cout<<"enter your
    age";
    std::cin >> vot.age;
    if (vot.age>=18)
    std::cout<<"you can
    vote";
    else
      std::cout<<"you
      can't not vote";
}
```

What is loop ?

A loop statement allows us to execute a statement or group of statement multiple times. ~~and following is the gener~~ We have three types of loops :- * For loop * While loop * Do while loop.

To run the body continously until a required condition is fullfill it is called looping.

Are you lazy to type std
use this :-

```
using namespace std;
```

If you want to use another ~~or too many~~ conditional use else if ( ). ~~In a conditional if = 1, else = 1. but else if = infinite.~~

```
int main () {
    int a=15, b=10;
    if (a==10)
    cout<<a<<"a is 10";

    else if (a==5)
    cout<<a<<"is 5";

    else
    cout<<"all is wrong";
}
```

if (a==10)
write a one statement under a conditional ~~if and else~~ is correct. write a one or two is incorrect.

If you want to write a two or more statement use block like this :-

```
if (a==10)
{
cout<<a<<"a is 25";
cout<<a<<"a is 10";
}
```

~~nesting of if is possible~~ You can nesting in conditional and loop.

If you want to use while loop you want to use while Keyword.

~~and~~ greater symbol is $\geq$.	while ( ) { } (conditional)	= this syntax of while

```
int main()
{
int a=25;
while (a>5)
{
```

15. difference between loop and function; see Y and Q

(--) is a decrement operater.

```
cout<<a<<"\n";
    a--;
    }
}
```

If you use while use loop your function. you need to call it 25 times.

```
while (a>5)
{
    cout<<a<<"\n";
    a--;
}
```

Check 24 is greater than 5

Print 24 and \n

loop still

25-1=24

Modify the conditional value

```
while (a>5)
{
    cout<<a<<"\n";
    a=a-5;
}
```

a=a-5; means 25-5

If you donot set Modification conditional like this

```
while (a>5)
{ cout<<a<<"\n";
}
```

a=a-5; is Modification conditional To stop loop

It will cout a infinite ~~to~~ and struck.

do while loop

If you want to use this loop. You want to use do the keyword.

Syntax of do while loop.

```
do {

} While ( );
```
Conditional

While loop and do while loop difference :-

While loop :- If the conditional is correct loop will done. the conditional is incorrect loop will not done.

do while loop :- The conditional is correct or incorrect the loop will done for 1 time.

```
int main () {
    int a=25;
    do {
        count<<a<<"\n";
        a=a-5;
    } while (a>5);
```

For loop

If you want to use this loop. You want to use For the Keyword.

Syntax of For loop

```
For ( ; ; )
{
}
```

(first ; → inclusion or declare; second ; → Condition;) → Modification)

```
int main () {
    int a1;
    For (a1=5; a1>0; a1=a1-2)
    {
        cout<<a1<<"\n";
    }
}
```

```
int main ()
{
  int a1;
  For (a1=0; a1<=10; a1++)
  {
    cout<<a1<<" ";
  }
  cout<<"done";
}
```

```
For (a1=10; a1>=0; a1--)
{
  cout<<a1<<" ";
}
```

How to print pattern like this.

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

```
int out;
For (out=1; out<=5; out++)
{
  int in;
  For (in=1; in<=out; in++)
  {
    cout<<in<<" ";
  }
```

```
cout<<"\n";
    }
```

out - row
in - column

```
int out;
for(out=5; out>=1; out--)
  {
    cout<<in<<"\n";
    int in;
    for(in=5; in>=out; in--)
      {
        cout<<in<<" ";
      }
cout<<"$";
```

any symbol

If you want you can to print. Symbol this is the way to print.

```
cout<<"\n";
}
```

≤ less than

≥ Greater than or equal to

<= less than or equal to

Switch

What is switch? It is the statement that allows a variable to be tested against a list of values for equality. The value in the switch is termed as a case, and hence the variable being switched on is checked against the case.

```
int main()
{
    char ans;
cout << "what is capital of
India \n";
cout<<"press a for new delhi \n";
cout <<"press b for Tamil nadu";
cin >> ans;
switch(ans) {
case 'a': cout << "correct";
break;
case 'b': cout<<"wrong";
break;
```

Why we use break?
If we not use break the next case will work

out put
correct wrong.
~~error~~

}
}

```
switch (ans) {
    case 'a': cout << "correct";
    break;
    case 'b': cout << "wrong";
    break;
    default: cout << "your
    entered wrong choice";
}
```

output

c

your entered wrong choice.

default used for if user entered wrong. It will cout the given text.

What is Static code Analysis?

Static program analysis is the analysis of computer software that is performed without actually executing programs, in contrast with dynamic analysis, which is analysis performed on programs while they are ~~executing~~ executing.

intro to PVS-Studio;

PVS-Studio is a tool for detecting bugs and security weakness in the source code of programs written in C, C++, C# and Java. It works under 64-bit systems in windows, linux and macOS environments, and can analyze source code intended for 32-bit, 64-bit and embedded ARM platforms.

goto statement

```
int main() {
    int a = 5;
    goto skip;
    a = a + 5;
skip: cout << a;
}
```

goto is the Key word

skip is a Variable You can use any Variable you want.

- avoid the between lines

output

5

Why 5 is cout? Because when you use you use goto. It skip the between ~~to the~~ lines to the Variable.

What is Random?

The rand() function in C++ is used to generate random numbers ; It will generate the same number every time we run the program. In order to seed the rand() function, srand(unsigned int seed) is used. The srand() function sets the inital point for generating the pseudo-random number.

```
int main() {
    int v1 = rand()%100;
    int v2 = rand()%100+1;
    int v3 = rand()%30+1985;
    cout << v3;
}
```

v1 in the range 0 to 99

v2 in the range 1 to 100

v3 in the rang. 1985-2014

When You use rand(). It will generate the same ~~number~~ random number every time. so we use .Srand().

time(Null) return the number (after conversion) of seconds since about midnight 1970-01-01. That number changes every second, so using the number to "select a book". pretty much guarantees a new sequence of "random" numbers every time your ~~prog~~ program runs.

```
int main() {
    Srand(time(NULL));
    int v1 = rand()%100;
    int v2 = rand()%100+1;
    int v3 = rand()%30+1985;
    cout << v3;
}
```

If you need to generate a random number between number to number. do like this :-

```
int main() {
  srand(time(Null));
  int max=6, min=1;
  int S = rand()%
  (max-min+1)+min;
  cout<<S;
}
```

arrays

What is arrays?

An arrays is a collection of elements of the same type placed in contiguous memory locations that can be individually referenced by using an index to a unique identifier. Five values of type int can be declared as an array without having to declare five different

Variables (each with its own Identifier).

int a1[ ] = { };

this two things are [ ] is declaring an array.

{ } is declaring an object.

a1 =

10	20	30	40	50
0	1	2	3	4 ...

this the starting position of a array.

Computer will ~~stro~~ store you value as 0 to...

How to use array.

you can only use with position without a position it will be error.

```
int main() {
    int a1[ ] = {10,20,30,40,50};
    cout << a1[0];
}
```

You number your array position.

```
int main ()
{
  int a1[] = {10,20,30,40,50};
  cout<<a1[0]+a1[3];
}
```

```
int main ()
{
  char b1[] = ("abc");
  or ('a','b','c'); are two type
  of store a char array.
  cout << b1[2];
}
```

("abc") without use a(,).
It work correct like given a(,). A ~~sing~~ single characters change to a index.

b1 = a b C
↓ ↓ ↓
0 index. 1 index. 2 index.

int a[50]

You can given value.

Size. you can store 50 or less 50 numbers or characters.

How to get input in array

```
int main ()
{
  int number [5];
  For (int i=0; i<5; i++)
  {
    cin >> number [i];
  }
  Cout << "this is output:"
  << number [3];
}
```

```
int main()
{
  int number [3];
  int sum = 0;
```

For (int i = 0; i < 3; i++) {

cin >> number[i];

sum = 0+2 sum = sum + number[i];

= 2, 2+4=6,

6+1=7.

}

cout << "answer is :" <<

sum = 2.

sum;

sum = 6.

sum = 7.

and the final sum is print.

Manipulating data

What is data?

There are three data types in C++ which are primitive data types, ~~and der~~ abstract data types, and derived data types. Primitive ~~dd~~ data types include integer, floating-point, character, boolean, double floating-point, valueless or Void and wide character.

* Strings :-

Alphabets, numbers, special characters ...

* Numbers :-

* int
* float
* digits

* Manipulate Null-Terminated strings. C++ supports a wide range of functions that manipulate null-terminated strings. These are :-

* strcpy (str1, str2): copies string str2 into string str1.

* strcat (str1, str2): concatenates string str2 onto the end of string str1.

* strlen (str1): Returns the length of string str1.

* strcmp (str1, str2): Returns 0 if str1 and str2 are the same; less than 0 if str1 < str2; greater than 0 if str1 > str2.

* strchr(str1, ch): Returns a pointer to the first occurrence of character ch in string str1.
* strstr(str1, str2): Returns a pointer to the first occurrence of string str2 in string str1.

** If you are using function of you should include :- # include<cstring>.

Strcpy :-

```
int main()
{
  char s1[50]={"hi"};
  Char s2[50]={"hello"};
  cout<<strcpy(s1,s2);
}
```

strlen :-

```
int main() {
char s1[50] = {"hi"};
char s2[50] = {"hello"};
cout << strlen(s1);
}
```

output

2

strcmp :-

```
int main() {
char s1[50] = {"hi"};
char s2[50] = {"hello"};
cout << strcmp(s1, s2);
}
```

operators used for string objects

* =: assignment
* +: Concatenation
* ==: Equality
* !=: inequality
* <: less than
* <=: less than or equal
* >: Greater than
* >=: Greater than or equal
* []: subscription
* <<: output
* >>: input

subscription is

char a = [ ]

a[0] = you enter value like hors.

```
int main() {
    char s1 = 's';
    char s2 = 's';
    cout << s1+s2;
}
```

```
int main() {
    int to b = 108;
    cout << char(b);
}
```

output

ss

Important functions supported by string class.

* append(): This function appends a part of a string to a another string.

* assign(): This function assigns a partial string.

* at(): This function

obtains the character stored at a specified location:

* begin (): This function returns a reference & to the start of the string.

* capacity (): This function gives the total element that can be stored.

* compare (): This function compares a string against the invoking string.

* empty (): This function returns true if the string is empty.

* end (): This function returns a reference to the end of the string.

* erase (): This function removes character as specified.

* find (): This function searches for occurrence of a specified substring.

* length(): It gives the size of a string or the number of elements of a string.

* swap(): This function swaps the given string with the invoking one.

append():-

```
int main()
{
 string str1 = "Mango is my
 favorite";
 string str2 = "Fruit";
cout << "Before appending, string values :"
<< str1 << '\n';
str1.append(str2,0,6);
cout << "After appending, string value
is :" << str1 << '\n';
 return 0;
}
```

Syntax

consider string str1 and str2. Syntax would be:-

* str1.append(str2);
* str1.append(str2, pos, len);
* str1.append(str2, n);

str: string object which is to be appended in another string object.

```
int main()
{
    string str1 = "Kashmir is nature";
    str1.append("of beauty", 10);
    cout << "String value is :" << str1;
    return 0;
}
```

Pos: It determines the position of the first character that is to be appended to another object.

len: Number of characters to be copied in another string object as substring.

Assign() :-

Syntax

consider two string str1 and str2,

Syntax would be: *str1.assign(str2);

~~str:str.~~ *str

*subpos: It defines the position of the character which is to be copied as a substring.

*sublen: It determines the number of characters of string to be copied in another string object.

*n

n: Number of characters to copy.

* ch: character value to be copied in times.

```
int main()
{
  string str = "javatpoint";
  string str1;
  str1.assign(str);
  cout << "Assigned string is :" <<
  str1;
  return 0;
}
```

another way

```
* str1.assign
(str, 7, 20);
cout << str1;
```

```
int main()
{
  string S;
S.assign(10, 'a');
cout << S;
  return 0;
}
```

```
int main()
{
  string S;
S.assign("javatpoint ", 10);
cout << "Assigned string is :" << S;
  return 0;
}
```

Syntax

Consider a string str. To find the position of a particular character, its syntax would be: *str.at(pos);

Pos: It defines the position of the character within the string.

```
int main()
{
string str = "welcome to C++";
cout<<"String contains!";
for(int i=0;i<str.length();i++)
  {
  cout<<str.at(i);
  }
cout<<'\n';
cout<<"String is shown
again";
```

```
for (int j =0; j < str.length(); j
    ++)
{
 cout << str[j];

 }
 return 0;

 }
```

find():-

```
int main()
{
string a1 = "raj"
if (a1.find("r"))
 cout << "found";
else
cout << "not found";
 cout << a1;

}
```

```
int main()
{
  string a1="raj";
  int s1= a1.find("a");
  if(s1!=string::npos)
   cout<<"found";
  else
  cout<<"not found";
}
```

List of Digits Manipulation

* reverse a number (for example input 1234, output: 4321).
* calculate sum and product of all digits.
* check number is prime or not.
* check number is palindrome or not.
* check number is Armstrong or not.

* Count digits in a numbers.
* print occurrence of a particular digit in a number.
* check number is perfect or not.
* check number is power of 2 or not
* check number is perfect square or not.

```
int main ()
{
  int a = 453;
  int last, first;
  last = a % 10;
  first = a / 10;
  cout << last << "\n";
  cout << first;
}
```

```
int main()
{
int a = 453;
int last=0, reverse=0, l1=0,
l2=0;
while(a!=0)
{
last = a%10;
reverse = reverse*10+last;
a=a/10;
}
cout<<reverse;
}
```

check !=10

change 3 to 5, next 4 comes

3 | 534 | 1

is stored in reverse

0+3=3, 3*10=30

1 2

30+5=35

350+4=354

3

The 2 numbers go already so there 0 conditional false.

Character is not an alphanumeric, isalnum() returns zero.

function	Description
int isalnum(int ch)	The isalnum() function returns nonzero if its argument is a letter or a digit. If the
int isalpha(int ch)	This function returns nonzero if ch is an alphabet, otherwise it returns zero.
int isdigit(int ch)	Returns nonzero if ch is a digit (ie., 0-9) otherwise it returns zero.
int islower(int ch)	Returns nonzero if ch is a lowercase letter; otherwise it returns zero.
int isupper(int ch)	This function returns nonzero if ch is uppercase; otherwise it returns zero.
int toupper(int ch)	Returns the uppercase equivalent of ch if ch is a letter; otherwise, char is returned unchanged.
int tolower(int ch)	Returns the lowercase equivalent of ch if ch is a letter; otherwise ch is returned unchanged.

```
int main ()
{
  Char b = 'h';
  Cout << char(toupper(b));
  return 0;
}
```

Storage classes and pointers.

Types of storage classes :-

* Automatic or local
* Register
* Static
* External
* Mutable

int a and like this are store under the Auto. You use int a or Auto a.

Difference between Ram and cache memory?

The cache is faster, cheaper, and smaller than Ram. The cache memory caches CPU data and programs often and quickly. Ram contains the CPU's current data and applications. The cache caches CPU-needed data and programs.

different between Auto and register.

* Auto are stored in normal memory
* Register are stored in cache memory

register int B register variable.

Keyword data types

Disadvantages of ch cache :- Cache ~~it the sat~~ have small memory so you can not store more Variables. So You can store important Variables.

Auto and register assign garbage value of our not gived value of the variable.

static and Extern are using Global memory.

static int C variable

Keyword data types

extern int D variable

Keyword data types

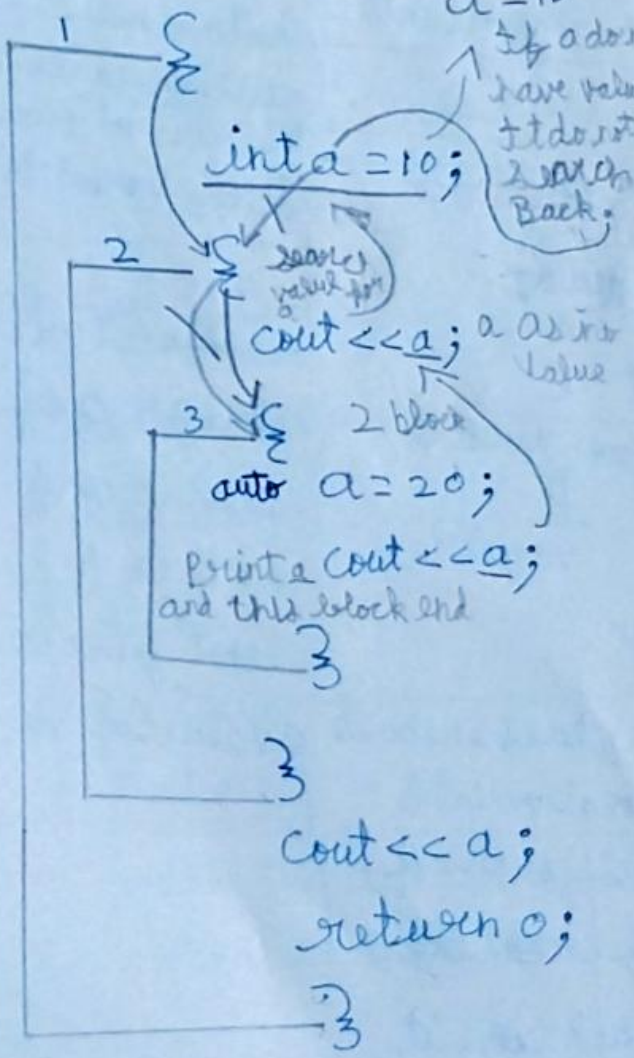
~~assign~~ It will assign 0 of static and extern If we donot give any value of variable.

The static storage class instructs the compiler to keep a local variable in existence during the life-time of the program instead of creating and destroying it each time it comes into and goes out of scope.

Therefore, making local variables static allows them to maintain their values between function calls.

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

int main()
{
    int a = 10;
    cout << a;
    {
        auto a = 20;
        print a cout << a;
        and this block end
    }
    cout << a;
    return 0;
}
```



The extern storage class is used to give a reference of a global variable that is visible to all the program files. When you use extern the variable cannot be initialized as all it does is point the variable name at a storage location that has been previously defined.

Small block will ~~ex~~ run first

If we use static a vari = 10. It value will fixed through Program. ~~token~~ Static variable will be fixed value.

The mutable storage class specifier is used only on a class data member to make it modifiable even though the member is part of an object declared as const.

You cannot use the mutable specifier with names declared as static or const, or reference members.

auto class :-

```
void autostorageclass()
{
cout<<"Demonstrating auto class\n";
int a = 32;
float b = 3.2;
Char c = "Greeks for Greeks";
Char d = 'G';
Cout<<a<<"\n";
Cout<<b<<"\n";
Cout<<c<<"\n";
Cout<<d<<"\n";
}
```

```
int main ()
{
    auto storageclass();
    return 0;
}
```

extern class :-

```
int x;
void externstorageclass()
{
    cout << "Demonstrating
    extern class \n";
    extern int x;
    cout << "Value of the variable
        'X'"
        << "declared, as extern
        :" << x << "\n";

    x = 2;
    cout << "Modified value of the
        variable 'x'"
        << "declared as extern:
        \n"
```

```
<<x;
}
int main()
{
  externStorageclass();
    return 0;
}
```

static class :-

```
int static Fun()
{
  cout<<"For static variables:";
  static int count=0;
  count++;
  return count;
}
int nonstatic Fun()
{
  cout<<"For nonstatic Variables:";
  int count=0;
  count++;
  return count; }
```

```
int main()
{
  cout<< static fun ()<<"\n";
  cout<< static fun ()<<"\n";
  cout<< non static Fun()<<"\n";
  cout<< nonstatic fun ()<<"\n";
  return 0;
}
```

register class :-

```
Void registerstorage class()
{
  cout<<"Demonstrating register
        class\n";
  register char b='G';
  Cout<<"Value of the Variable 'b'"
    <<"declared as register:"<<
    b;
}
```

```
int main()
{
register storageclass();
return 0;
}
```

mutable class :-

```
#include<iostream>
using std::cout;
class Test {
public:
    int x;
    mutable int y;
    Test()
    {
      x=4;
      y=10;
    }
};
```

```
int main()
{
    const test t1;
    t1.y = 20;
    cout << t1.y;
    return 0;
}
```

Thread_local storage class:-

The thread_local storage class is the new storage class that was added in C++11. We can use the thread_local storage class specifier to define the object as thread_local. The thread_local variable can be combined with other storage specifiers like static or extern and the properties of the thread_local object changes accordingly.

Properties of thread_local storage class:-

* Memory location: RAM
* Lifetime: till the end of its thread.

```
thread_local int var=10;
int main()
{
  thread th1([]() {
    cout<<"thread 1 var value:"<<
    (var+=18)<<'\n';
  });
  thread th2([]() {
    cout<<"thread 2 var value
    :"<<(var+=7)<<'\n';
  });
  thread th3([]() {
    cout<<"thread 3 var value:
    <<(var+=13)<<'\n';
  });
  th1.join();
  th2.join();
  th3.join();
  return 0;
}
```

Pointers

* to get address of the value
* Symbols * and & & are used

```
int main ()
{
    int a;
    a = 10;
    int *p;
    p = &a;
    cout << p;
}
```

int → Data type, p → vari

change normal variables to pointer variables

extra content and extra explanation

variable :- * It is used to store data
* It is ~~used to~~ a name of memory location

~~endl~~ working is similar to the '\n' escape sequence.

C and C++ differences

C	C++
Developed by Dennis Ritchie in 1972	Developed by Bjarne stroustrup in 1980
Procedural oriented Programming language	object oriented Programming language
Data is not secure	Data is more secured
We can't use functions in structure	We can use functions in structure
scanf() and printf() is used for input and output purpose	Cin and cout is used for input and output purpose.

```
int main ()
{
  int ~~main~~ num 210; num1=20;
  Cout << num << endl << num1;
  }
```

```
int main ()
{
  int a=5, b=7;
  a+=b;
  Cout<<a;
  }
```

a+=b is a shortcut for a = a+b.

```
int main ()  Post
{
  int a=25, b=5;
  a-=b;
  Cout <<a;
  }
```

a-=b is a shortcut for a = a-b.

(++)- increment
(--)- decrement

(++) a - pre increment or (--) or decrement

a(++) - post increment or decrement (--)

```
int main ()
{
  int a = 25;
```

```
cout<<a++;
cout<<a;
}
```

Post i++ will not ++ in that line. It will ++ in next line but pre ++ will be ++ in that line.

```
int main()
{
int a=25;
cout<<++a;
}
```

pre

```
int main()
{
int a=5,b;
b=a++ + ++a;
cout<<b
<<endl<<a;
}
```

1 2 3 4 5

5+1=6

a=6, a=6

6+6=12

b=12

a=6+1=7

* Pre inc / dec - 1 step
* substitution - 2 step
* evalvation - 3 step
* assignment - 4 step
* post int/dec. - 5 step.

(!=) not equal.

if(!(a>40 && a<=90))

Logical operator -
* and (&&)
* or (||)
* not (!) If true it change to false If false it change to true.

conditional or ternory operator :-
>, ?, :
syntax:

(condition) ? value : value

If the condition true this will run

If the condition false this will run

```
int main()
{
  int a = 50;
  int x = 2;
  x = (a > 20) ? 20 : 30;
  cout << x;
}
```

Printing a table of Number

```
int main()
{
  int i, n = 5;
  for(i = 0; i <= 15; i++){
  cout << n << "*" << i << "=" << n*i
  << endl;
  }
}
```

Find odd and Even Numbers in 0 to number

Even

```
int main()
{
  int i;
  For (i=1;i<=50;i++){
  if (i%2==0)
  cout<<i<<endl;
    }
  }
```

odd

```
int main()
{
  int i;
For (i=1;i<=50;i++){
if (i%2!=0)
cout<<i<<endl;
  }
  }
```

```
int main()
{
  int i=0,j;
while(i<5){
  j=0;
while(j<5){
cout<<"*";
j++;
  }
cout<<endl;
i++;
}
```

output

```
}
void main()
{
  int i;
  For (i=1; i<=10, i++)
  {
    Cout<<i<<endl;
    Continue;
    Cout<<"Welcome";
  }
}
```

this line only work next lines will not work. See the continue output.

If we use break it exit for the loop. See the break output.

continue output

1
2
3
4
5
6
7
8
9
10

break output

1

```
void main()
{
  int i;
  For (i=1; i<=10; i++)
  {
```

```
if (i==6)
{
    continue; → If you use break. output will be 1 2 3 4 5 break output
}
cout<<i<<endl;
}
}
```

continue output

	break output
1	1
2	2
3	3
4	4
5	5
7	
8	
9	
10	

```
int main()
{
    int a,b;
    string ans-;

    Cretturonum:
    cout<<"Enter two numbers:";
    cin>>a>>b;
    if (a>b)
        cout<<"A is greater"<<endl;

    else if (b>a)
        cout<<"B is greater"<<endl;

    else
        cout<<"Both are Equal"
        <<endl;
```

```
cout << "Do you want to continue
? Yes / no " << endl;
cin >> ans-;
if (ans == "yes")
{
  goto Gettwonum;
}
}
```

```
int main ()
{
  int x[5] = {1,2,3,4,5};
  for (int i=0; i<5; i++)
  {
    cout << x[i] << endl;
  }
}
```

```
int main ()
{
  string s[5] = {"raj", "hi",
  "shoot", "cat"};
```

```
For (int i=0; i<s; i++)
  {
    cout << s[i] << endl;
  }
  }
```

Reference variable

```
int main()
{
  int x = 10;
  int & y = x;
  cout << x << endl << y;
}
```

What name you want You can give like x, z etc

Reference variable

```
~~int~~ int main()
{
  int x=10;
  int *p = &x;
  cout << p << endl;
  cout << *p;
  }
```

output

0X503270 address

10 value in the address

Continue in Theory and questions Page no: 12.

. is a special program
a programming language's
machine code, bytecode or
anguage.

derstoode to
hat are created
or a
ed in a file.
it, a
vice

THEORY
And
Questions

ser
of the
cation and
ge.
guage is not
not be understood
nterpreter.

lock of code that performs
. A function can optionally
parameters that enable callers to
uments into the function. A function can

```
for (int i=0; i<5; i++)
  {
    cout<<s[i]<<endl;
  }
  }
```

Refer

in

give

x, z

int

```
{
  int x=10;
  int *p=&x;
  cout<<p<<endl;
  cout<<*p;
  }
```

Continue in Theory and questions Page no: 12.

1. A Compiler is a Special Program that translates a programming language's source code into machine code, bytecode or another programming language.

2. Source code is generally understoode to mean programming statements that are created by a programmer with a text editor or a visual programming tool and then saved in a file. Object code generally refers to the output, a compiled file, which is produced when the source code is compiled with a compiler.

3. Syntax refers to the rules that define the structure of a language. Syntax in computer programming means the rules that control the structure of the symbols, punctuation and words of a programming language.

If the syntax of language is not followed, the code will not be understood by a compiler or interpreter.

4. A function is a block of code that performs some operation. A function can optionally define input parameters that enable callers to pass arguments into the function. A function can

optionally return a value as output.

5. Keywords are those words whose meaning is ~~already~~ already defined by compiler. These Keywords cannot be used as an identifier. Note that Keywords are the collection of reserved words and predefined identifiers. Predefined identifiers are identifiers that are defined by the compiler but can be changed in meaning by the user.

6. "Caller" refers to the function that calls the function you are thinking about. So in your example, main() is the caller, and this function() is the "callee" (or, the function being called).

7. A function prototype is a declaration of the function that informs the program about the number and kind of parameters, as well as the type of value the function will return. One incredibly helpful aspect of C++ functions is function Prototyping.

8. The values that are declared within a function when the function is called are known as an argument. These values are

considered as the root of the function that needs the arguments. While execution, and it is also known as Actual arguments or Actual Parameters.

9. The Values that are declared within a function when the function is called are known as an argument. Whereas, the variables that are defined when the function is declared are known as a parameter.

10. These are those files that store ~~de~~ predefined functions. It contains definitions of functions that you can include or import using a preprocessor directive #include. This preprocessor directive ~~#in~~ tells the compiler that the header file needs to be processed prior to the compilation.

11. Header File :-

A header file is the file where all the headers name are mentioned that going to be used or consumed in the main code file.

Library :-

A ~~is~~ library is the file where the implementation code of each header is written down which is mentioned in the Header file.

12. Functions are reusable codes that perform a specific ~~task~~ task when they are called. Derived data types are formed from fundamental data types. Structures are one such user-defined data type. The structures can have many fundamental data types known as structure members grouped into a single user-defined data type.

13. Structures (also called structs) are a way to group several related variables into one place. Each variable in the structure is known as a member of the structure. Unlike an array, a structure can contain many different data types (int, string, bool, etc.).

14. int main (void) represents that the function takes No argument. Suppose, if we don't keep void in the bracket, the function will take any number of arguments. The syntax of int main (void) is as follows:- int main (void) {..... return 0;}.

15. Just as a loop is an embodiment of a piece of code we wish to have repeated, a function is an embodiment of a piece of code that we can run anytime just by calling it into action.

1. Some commonly used keywords in C++

asm	double	operator	this
auto	else	private	throw
break	enum	protected	try
case	extern	public	typedef
catch	float	register	union
char	for	return	unsigned
class	friend	short	virtual
const	goto	signed	void
continue	inline	sizeof	volatile
default	int	static	while
delete	long	struct	if
do	new	template	switch

*asm : To declare that a block of code is to be passed to the assembler.

*auto : A storage class specifier that is used to define objects in a block.

* break : Terminates a ~~stat~~ switch statement or a loop.

* case : used specifically within a switch statement to specify a match for the statement's expression.

* catch : Specifies actions taken when an exception occurs.

* char : Fundamental data type that defines character objects.

* class : To declare a user-defined type that encapsulates data members and operations or member functions.

* const : To define objects whose value will not alter throughout the lifetime of a program execution.

* continue :- Transfers control to the start of a loop.

* default :- Handles expression values in a switch statement that are not handled by case.

* delete : Memory deallocation operator.
* do : indicate the ~~stra~~ start of a do-while statement in which the sub-statement is executed repeatedly until the value of the expression is logical-false.
* double : Fundamental data type used to define a floating-point number.
* else : Used specifically in an if-else statement.
* enum : To declare a user-defined enumeration data type.
* exter : An identifier specified as an extern has an external ~~likage~~ linkage to the block.
* float :- Fundamental data type used to define a floating-point number.
* for : indicates the start of a statement to achieve repetitive control.
* friend : A class or operation whose * implementation can access the private data members of a class.

* goto: Transfer control to a specified label.
* if: indicate the start of an if statement to achive selective control.
* inline: A function specifier that indicates to the compiler that inline substitution of the function body is to be preferred to the usual function call implementation.
* int: Fundamental data type used to define integer objects.
* long: A data type modifier that defines a 32-bit int or an extended double.
* new: Memory allocation operator.
* operator: overloads a C++ operater with a new declaration.
* Private: Declares class members which are not visible outside the class.
* Protected: Declares class members which are private except to derived classes.

* public : Declares class members who are visible outside the class.
* register : A storage class specifier that is an auto specifier, but which also indicates to the compiler that an object will be frequently used and should therefore be kept in a register.
* return : Returns an object to a function's caller.
* short : A data type modifier that defines a 16-bit int number.
* signed : A data type modifier that indicates an object's sign is to be stored in the high-order bit.
* sizeof : Returns the size of an object in bytes.
* static : The lifetime of an of object-defined static exists throughout the lifetime of program execution.
* struct : To declare new types that encapsulate both data and member functions.

* Switch : This Keyword is used in the "Switch statement".
* template : parameterized or generic type.
* this : A class pointed points to an object or instance of the class.
* throw : Generate an exception.
* try : indicates the start of a block of exception handlers.
* typedef : Synonym for another intergral or user-defined type.
* Union : Similar to a structure (struct) in that it can hold different types of data, but a union can hold only one of its members at a given time.
* Unsigned : A data type modifier that indicates the high-order bit is to be used for an object.

* Virtual: A function specifier that declares a member function of a class that will be redefined by a derived class.
* void: Absent of a type or function parameter list.
* Volatile: Define an object which may vary in value in a way that is ~~unde~~ undetectable to the compiler.
* while: Start of a while statement and end of a do-while statement.

2. Types of Escape sequences.

Escape Sequence	Represents
\a	Bell (alert)
\b	Back space
\f	Form Feed
\n	new line
\r	carriage return
\t	Horizontal tab
\v	Vertical tab
\'	Single quotation Mark
\"	Double quotation Mark
\\	Backslash
\?	Literal question Mark
\ooo	ASCII character in octal notation
\x hh	ASCII character in hexadecimal notation
\x hhh	

Unicode character in hexadecimal notation if this escape sequence is used in a wide-character constant or a Unicode string literal.

```
int add (int x, int y)
{
  cout << x+y << endl;
  return x+y;
}
int
void main()
{
  int c;
  c = add(10,15);
  cout << "The value of c is :" << c << endl;
}
```

int if we use void this will error

```
float add (int x, int y)
{
  float z = (float) x/y;
  return z;
}
int main()
{
  float c;
  c = add(10,20);
  cout << "The value of c is :" << c << endl;
}
```

functions :-
call by value

```
int add (int x, int y)
{
  int z = x + y;
  return z;
}

int main()
{
  int a, b;
  a = 5, b = 5;
  cout << add (a, b) << endl;
}
```

This a way for call by value

This also a way for call by value

functions :-
call by reference

```
int add (int x, int y, int &z)
{
  z = x + y;
  return z;
}
```

```
int main()
{
    int a=5; int b=5; int c;      passes c value to the c
    Cout<< add(a,b,c) << endl;
    Cout << "The value of c is :"<<c;
}
```

OOP's :- Object Oriented Programming

~~System.~~

What is OOP?

OOP stands for object-oriented programming. Procedural programming is about writing Procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

Features of OOP's :-

* Classes and objects
* Abstraction
* Encapsulation
* inheritance
* Polymorphism
* Message Passing
* Dynamic Binding

~~Features of oops :-~~

What is Class ?

A class in C++ is a user-defined type or data structure declared with keyword class that has data types and functions as its members whose access is governed by the three access specifiers private, protected or public.

What is object ?

In C++, object is a real world entity, for example, chair, car, pen, mobile, laptop etc... In other words, object is an entity that has state and behavior. Here, state means data and behavior means functionality. object is a runtime entity, it is created at runtime. object is an instance of a class. All the members of the class can be accessed through object.

class & calculator Variable for your class or name

{ Keyword

Why we use Public: because members are in private.

public:

int a; int b; - Data members

int add() - Member Function

{

return a+b;

}

};

int main()

{

calculator calc1; for your object.

↳ name of your class. Name of

calc1.a = 10;

calc1.b = 20;

cout << calc1.add();

}

structure:

* structure is a user defiend Data type.

* It is a collection of heterogenous variable.

C structure :-

* We can declare only variables i.e., Data members.
* Structure members by Public.

C++ structure :-

* We can declare both variables and member function.
* Structure members can private / public / Protected.

C++ structure	C++ Class
By default, structure members are public.	By default, class members are private.
Cannot be inherited	can be inherited.
It is useful to ~~bu~~ build simple programs.	It is useful to ~~bu~~ build complex programs.

```
class employee
{
  Public:
        int id;
        string name;
        float salary;
        void print();
    };
```

```
void employee :: print()
                 Scope Resolution
{
  cout<<id<<"."<<name<<salary<<endl;
}
int main(){
    employee e;
    e.id = 1;
    e.name = "raj";
    e.salary = 20000;
    e.print();
}
```

Access Specifiers

* Public
* Private
* Protected

* Public - Access from Anywhere in a program.
* Private - Access only by the member function of the same class and friend function.
* Protected - It is same link private one additional benefit that they can be accessed in child classes.

```
class bank
{
  private:
        double balance = 0;
  Public:
        string acctname;
        string acctnumber;
    double checkbalance()
    {
      retevrn balance;
      }
      Void credit (double amt)
      {
        bb balance += amt;
          }
      void dd debit (double amt
        {
          balance -= amt;
            }
              };
```

```
int
void main()
{
  Bank A Acct1;
  Acct1.acctname = "Hari";
  Acct1.acctnumber = "5107";
  Cout << Acct1.checkbalance() << endl;
  Acct1.credit(10000);
  Cout << Acct1.checkbalance() << endl;
  Acct1.debit(500)
  Cout << Acct1.checkbalance();
}
```

Getters and Setters

```
Class Rectangle
{
  Private:
  int length, width;
  Public:
  int getlength()
  {
    return length;
  }
};
```

```
void setlength (int x)
{
  length = x;
    }
  int getwidth()
  {
    return width;
      }
    };
  void setwidth (int y)
    { {
        width = y;
          }
            };
int main() {
  rectangle box;
  int len, wid;
  Cout << "enter length and width of rectangle";
  Cin >> len >> wid;
  box.setlength (len);
  box.setwidth (wid);
  Cout << "Area of rectangle is :" << box.
  getlength() * box.getwidth();
  }
```

Constructors

* Constructors Name Could be the name of should be same Class and should be as class name. Public etc.
* It should be public.
* It doesnot contains any return type.
* It will be called only once while creating an object.
* It may contain parameter which is called parameterized constructor.

Class Rectangle Default constructor

```
{
private:
    int length;
public:
    Rectangle()
    {
    length = 0;
    }
    int getlength()
    {
    return length;
    }
};
```

```
void main ()
{
  Rectangle box1;
  Cout << box.getlength() << endl;
    }
```

Parameterized constructor.

```
Class Rectangle
 {
  Private:
        int length;
   Public:
     Rectangle (int x)
     {
       length = x;
         }
       int getlength ()
       { return length;
           }
             };
void main () {
  Rectangle box1 (10);
  Cout << box.getlength() << endl;
                    }
```

copy constructors:-

A copy constructors is a member function that initializes an object using another object of the ~~Sa~~ same class.

```
class box
{
  int length, width;
 Public:
       Box ()
       {
        length = 0;
        width = 0;
         }
       Box (int x, int y)
       {
        length = x;
        width = y;
         }
       Box ( const Box & oldbox)
       {
        length = oldbox.length;
        width = oldbox.width;
       }
```

so Const we use we can't change the Box data.

If we can't use const we can change the length and width of Box 1 and Box 2 etc.. Box.

Continue in computer language 2.