## Agenda-

```
Constructor & its Types
Constructor Members Initializer List
Destructor
Mutators / Inspectors
Modular Approach
Constant
Reference
```

## Constructor & Types

```
Parameterized Ctor-
If you want to create an object by passing some values you need to write a
parameterized ctor.
 Complex(int real, int imag){}
```

## Destructor

```
Dtor is also a special member function of a class
Its name is same as that of class with ~(tild) operator
It is automatically called.
Dtor calling sequence is exactly opposite to Ctor calling sequence
```

## Mutators

```
Mutators are also called as setters
These are generally used to modify the state of the object
```

## Inspectors

```
Inspectors are also called as Getters
These do not modify the state of object.
```

## Constant

```
1. Datamember
    If we keep data members as constant then they must be initialized only in Ctor
```

```
members initializers list.
2. Member Functions
    If we dont want to modify state of any data member then we must make the
function as constant.
    If we still want to modify the non const data member inside const member
function make that datamember as mutable.
3. Object
    If we make object as constant then we cannot call non constant member
functions.
```

# Reference

```
It is similar to typedef.
It is like an alais to your variable.
int num1;
int &ref = num1;
```

## In C++ we can pass argument to the function

```
1. By value
2. By address
3. By reference
```