Day 3

DATE

Dynamic Constructor

- constructor can allocate dynamically created memory to the object.
- Thus, object is going to use memory region, which is dynamically created by constructor.

eg:
```
complex ()                 → constructor.
{
    p = new int;           → a new block is created dynamically.
    *p = 5
}
```

Namespaces

- Namespace is a container for identifiers.
- It puts the name of it's members in a distinct space so that they don't conflict with the names in other namespaces or global namespace.

- How to create namespace ?

```
namspace rucha {
        // declarations
}
```

- ~~Dose~~ Doesn't end with ';'
- It must be of global scope or nested with other namespace

DATE

- you can use alias name for your namespace name for ease of use.

    namespace rucha = ru;

    (Now, instead of using 'rucha' you can use 'ru' also)

- Namespace is not a class. You can't create an instance
- Namespaces can be unnamed also.
- A namespace definition can be continued and extended over multiple files, they are not redefined or overridden.

· Example

```
# include <iostream>
  using namespace std;
  namespace rucha
{       int a;
        int f1 ();
        Class A
     {
        public:
             void f1 ();
     };
 }                                   Declaration
                  rucha ::
                  ~~myspace::~~
 int f1 ()
{  cout << "Hello f1";           ---->  Defn of function f1
}
```

~~using namespace myspace;~~
~~int main()~~
~~{ a= 5;~~

```
void ~~mys~~ rucha :: A :: fun1 ()
{     cout << " Hello fun1 " ;
}
```

$\longrightarrow$ Def^n of function fun1

```
using namespace rucha ;

int main ()
{
    ~~int~~ a = 5 ;
        f1 () ;
        A obj ;
        Obj . fun1 () ;

}.
```

Nested class

- Class inside a class is called nested class.
- A nested class is a member and as such has the same access rights as any other member.
- The member of an enclosing class have no special access to members of a nested class; The usual access rule shall be obeyed.