Date:.....

ITEM 4

→ Always initialize objects before using them.

```
int x=0;  // manual initialization
double d;
cin>>d;  // initialization by reading from
         i/p stream
```

⇒ Assignment & initization are different.

```
A::A(const string &name, const string &add)
{
  theName = name;
  theAddress = add;
  num = 0;
}
```

```
A{
private:
  string theName;
  string theAddress;
  int num;
};
```

Initialization is :-

```
A::A(const string &name, const string &add)
  : theName(name), theAddress(add),
    num(0)
{ }
```

Above one is the better way to write ctor.

Date:.....

⇒ The assignment first calls default Ctor to initialize name, theAddress & then assigns new value on top of default constructed ones.

⇒ The arguments in the initialization list like theName is copy constructed from name, theAddress is "copy" from add

Thus, single call to copy constructor is more efficient, than call to default constructor followed by call to Copy assignment operator.

Date:.....

Key Points.

=> Base classes are initialized before derived class.

=> Within a class, data member are initialized in the order in which they are declared.

=> Static object exist from time it is constructed till the end of program.

=> Reference do not refer to different object.

=> Pointer may point to any memory location but reference once pointed cannot be changed.

non-local static objects → object outside functions ~~outside~~

```
Class FileSystem {
   public:
      size_t num() const;
};

extern FileSystem tfs; // static non-local object

Class Directory {
   public:
      Director (params);
   { size_t disks = tfs.num();
   }
};
```

Filesystem & Directory are two different source files.
Unless tfs is initialized, Directory's object uses tfs, hence ~~cannot~~ undefined.

→ Best way to solve this is ~~call static~~ make non-local static object as static local object.

Date:.....

```
class FileSystem
  {};
  FileSystem &tfs()
    { static FileSystem fs;
      return fs;
    }
class Directory {};
Directory: Directory (params)
  { size_t disks = tfs().numd();
  }
  Directory &tempDir()
    { static Directory td;
      return td;
    }
};
```

The above functions return references to objects they contain → FileSystem &tfs() {}

Clients call the funct instead of referring to objects.

↓ funct.

Eg → tfs().num()
instead of tfs.num();