Name: Atreya Sawani, Prastav

Access Control:

```
class Abc {
    Public:        //Access Specifier
        Abc(int)
        Abc();
        int &x(int);
    Private:       //Access Specifier
        int -x;
};
```

Classes:

```
Class Abc{        //class
    Public:
        int x;
};
```

~~Class Scope:~~

Constructor function:

```
class Abc{
    Public:
        Abc(int);   //Constructor
        Abc();
        int &x(int);
}
```

Data Members:

```
class Abc {
    Public:
        Abc(int);
        Abc();
        int &x(int);   → member of Public data member
    Private:
        int -x;  → Private data member
};
```

Default arguments:

```
int Sum( int x, int y, int z=0)
{   return x+y+z;
 }
int main()
 {
    Sum(5,10,15);
    Sum(5,10);
    Sum(1
 }
```

Dynamic Allocation:

```
int *y;
 y =new int;
double *x;
 x =new double;
```

Encapsulation:

```
Class  Abc {
   Private:
      int x;
    Public:
       Void set (int y)
       {
          x = y;
       }
       int  get()
       {
         return x;
       }
   };
int main()
 {
    Abc a1;        //encapsulates int x, Set() & get().
    a1.set(2);
 }
```

Overloading:

```
class Abc {
  Public:
     Void Hello(int x)
       {
        cout << "int" << x;
       }
     Void Hell (double x){
        Cout << "double" << x;
        }
};
```

Pointers:

```
int main() {
    int x;
    int y;
   Cout << "Address of x" << &x;
   Cout << endl;
   cout << "Adress of y" << &y;
}
```

Polymorphism:

1. function overloading
2. Operator overloading.

References:

```
int main(){
   int x = 1;
   int& ref = x;   //ref is reference to x
   ref = 2;        // x is changed to 2.
   x = 3;          //ref & x are both 3 now;
}
```

File scope:

```
static int n;      // File Scope Variable.
float f;           // global Variable
int main()
 { double d;       // local variable.
 }
```

Function Members:

```
class c                                  //function scope is within class
{ Public:
        int add()                        // function member of a class
         { return a+b; };
  Private:
        int a,b;
}
```

Inheritance:

```
class parent {
  Public:
     int x;
};

class child : Public parent
  { Public:
      int y;
  };
class grandchild : Puble Parent, Public child{
    Public :
       int z;
    }
```

NameSpaces:

```
using namespace std;
int main() {
   cout << "Hello";     // std:: cout << "Hello";
```

Scope:

1. Global scope
2. Local scope
3. Namespace scope
4. Class scope
5. Function scope

Structures:

```
Struct student{
    int id;
    String name;
};
```