Revisiting Pointers

Initialization list in Constructors

- Objects can be initialized by initialization list in constructors.

- Syntax:

```
   constructor (argument-list) : initialization-section
{
    constructor-body
;
}
```

- For a class Test having integer variables a and b, such that a is declared before b, initialization list in constructor can be created in the different ways:

i. `Test(int i, int j) : a(i), b(j)`

ii. `Test(int i, int j) : a(i), b(i+j)`

iii. `Test(int i, int j) : a(i), b(2 * j)`

iv. `Test(int i, int j) : a(i), b(a + j)`

v. `Test(int i, int j) : b(j), a(i+b)` → will throw error bcoz a is declared before b in the class

vi.
```
Test(int i, int j): a(i)
  {
      b = j;
   }
```

Dynamic Memory allocation: new and delete Keywords

1. new

- `int *p = new int(40);`
- `float *p = new float(40.78);`
- `int *arr = new int[3];`

```
arr[0] = 10;
*(arr+1) = 20;
arr[2] = 30;
```

→ Third line dynamically allocates the space of 3 integers in contiguous locations in memory and returns a pointer storing the address of first integer.

2. delete

`delete arr;` →It deletes that variable/object which is pointed by p.

`delete[] arr;` →dynamically deletes array which is pointed by p.

Pointers to Objects

```
class Complex{
      void setData(int a, int b){}
};
Complex c1;
Complex *ptr = &c1;
      //or
Complex *ptr = new Complex; →dynamically
```

Arrow Operator:

```
// (*ptr).setData(1, 54); is exactly same as
    ptr->setData(1, 54);
```

Array of Objects

```
class_name *ptr = new class_name[array_size];
```

→ Allocates memory for the objects of the class and returns a pointer pointing to the first object of the array.

→ Writing `ptr++;` will make the ptr pointing to the next object in the array.

→ Elements of the class can be accessed by arrow operator.

this Pointer

`this` is a keyword which is a pointer which points to the object which invokes the member function.

```
this->class_variable
```

→ It allows us to pass the arguments (to the member function) of same name as that of class variable.

→ It is also used to return the object from a member function by specifying the return data type as <class name &> for the function.

```
class_name & func(int arg){
    return this;
}
```