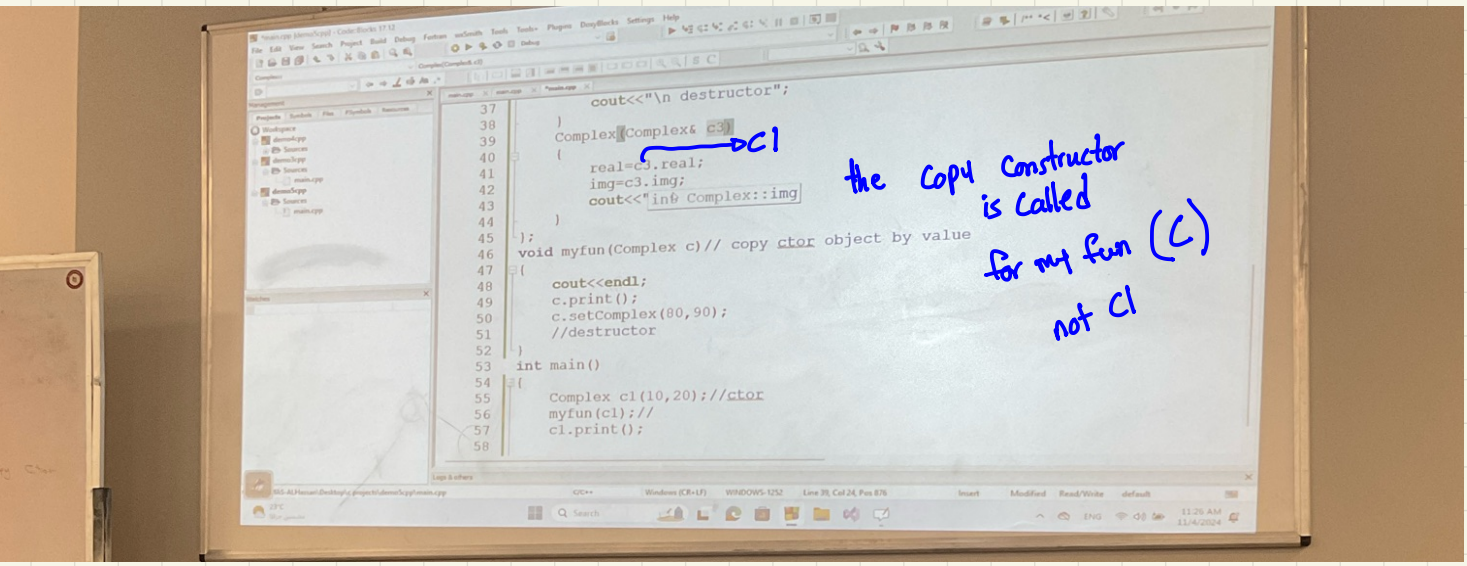
OOP Course

Created by : Israa Abdelghany
LinkedIn : www.linkedin.com/in/israa-abdelghany-4872b0222
GitHub : https://github.com/IsraaAbdelghany9

Lecture 3

- complex class
  - setters
    - img
    - real
    - complex
  - getters
    - Real
    - Img
  - print
  - Constructor
    - set one param (real)
    - set both img & real
  - Copy Constructor
  - destructor



c1 is c3

this → object member
↳ instance member

```
    ~Complex(){
        cout<<"\n destructor";
    }
    Complex(Complex& c3)
    {
        real=c3.real;
        img=c3.img;
        C3. img= 7000; C3. real= 5000;
        cout<<"\n copy ctor";
    }
};
void myfun(Complex c)// copy ctor object by value
{
    cout<<endl;
    c.print();
    c.setComplex(80,90);
    //destructor
}
int main()
{
    Complex c1(10,20);//ctor
    cout<<"\n address of c1"<<&c1;
```

```
45          c3.img=8000;
46          cout<<"\n copy ctor";
47      }
48  };
49  void myfun(Complex c)// copy ctor object by value
50  {
51      cout<<endl;
52      c.print();
53      c.setComplex(80,90);
54      //destructor
55  }
56  int main()
57  {
58      Complex c1(10,20);//ctor
59      cout<<"\n address of c1"<<&c1;
60      myfun(c1);//
61
62      c1.print();   => 5000 + 7000 j
63
64      return 0;
65  }
66
```

my fun create object by value so to initialize the copy of the function ⇒ copy constructor is called

creating object ⇒ copy constructor is used

after finishing myfun will call the destructor to destroy the copy which is c

if myfun (Complex & c) ⇒ no copy will be called
- alias created
- alias deleted after finishing myfun

myfun (c1)

if myfun (Complex * c)

```
c -> print();
c -> set Complex (2,3);
```

my fun (&c1)

Case 2 of Copy Constructor

```
int main()
{
    Complex c1(10,20);
    Complex c2(c1);
```

⇒ will call Copy Constructor

OR

```
    Complex c4 = c1;
```

⇒ will call Copy Constructor

Not like ⇒ `C4 = C1;` ⇒ no Copy Called

```
}
```

if my_fun (Complex* c3) ⇒ no destructor is called

```
Complex(Complex& c3)
{
    real=c3.real;
    img=c3.img;

    cout<<"\n copy ctor";
}
};
void myfun(Complex c,Complex r)// copy ctor object by value
{
  cout<<"\n myfun\n";
   //destructor
}
int main()
{
    Complex c1(10,20);
    Complex c4(80,90);
    myfun(c1,c4);

    return 0;
}
```

```
Complex::Complex()
Complex::Complex(Complex& c3)
Complex::Complex(int _real)
Complex::Complex(int _real, int _img)
class Complex {...}
```

2 Constructor
2 Copy Constructor
4 destructors

if I defined the Copy constructor I need to define its function myself or it will return garbage data to the object is copying data

```
Complex (Complex & c3)
{
    this->real = c3.real;
    img = c3.img
    cout << " in copy constructor " << endl;
}
```

Some notes: any thing in gray is a note not part of the code.

1. real = real of the c (the object copying the data)
2. c3 is c1 which is sent to the function.
3. this ⟹ is an instance pointer for c not for c1 or c3
4. if I commented real=real and img = img this will not affect the values of c3 but the copy created from c3 which is c will have garbage data.
5. I can access and change the data of c3 and change it like c3.setimg(3); c3.setreal(2); and this will also affect the c1 object

* Cases where Copy Constructor is called :-

1. object by value to function (not ref or *)
2. Complex C4(C1)
3. complex C4 = C1
4. Return object by value

```
C5 = myfun(C1, C2);
```

↳ after exiting myfun it's destroyed but saving the value in temp using Copy Constructor then saved in C5. then temp is deleted.

* all return save the value in a temp (work in a same way)

* I don't need to type copy constructor my self but I need to define it if there is a dynamic

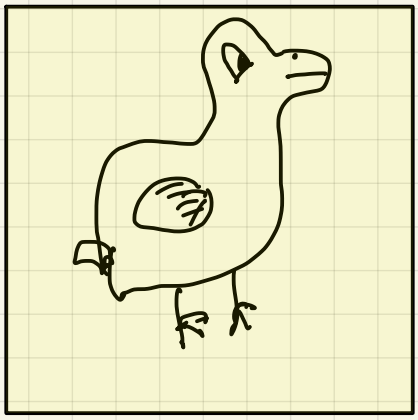
---

```
complex myfun(Complex C3)
{ Complex res;
} return &res ⟹ error
```

But if

```
Complex* myfun(Complex C3)
{ Complex * res = new Complex();
} return &res; ⟹ will not Call Copy Constructor and no error
                but not the value of res
```

Stack



LIFO

(Last in first out)

* to create that I need to use OOP to create restrictions on the usage and hide my array

* I need variable to keep tracking the current (last used) index tos (top of stack) initialized by -1
* pop will decrease the index and push will increase it and enter data
* tos range is -1 : (range - 1)
  - -1 : empty
  - (range - 1) : full

```
Class stack
{
    int arr[5];
    int size;
    int tos;
  Public:
    stack ()
    { tos = -1;
      size = 5;
    }
```

```
void Push (int d)
{
  if(tos < size -1)
    { tos++;
      arr[tos]=d;
    }
  else cout<<" stack is full ";
}

int pop ( int & d)
{  if (tos != -1)
    {
      d = arr[tos];
      tos --;
      return 1;
    }
  else
    {cout <<"stack is empty";
     return0;
    }
}
```

check again

```
Stack s1;
s1.Push(30);
s1.Push(40);
s1.Push(50);
s1.Push(50);
s1.Push(50);

int x=0;;
if(s1.pop(x)==1)
    cout<<endl<<x;//50
if(s1.pop(x)==1)
    cout<<endl<<x;//40
if(s1.pop(x)==1)
    cout<<endl<<x;//30
if(s1.pop(x)==1)
```

```
Class stack
{
    int size;
    int tos = -1;
    int arr[5];

  Public:
    stack ()
    { tos=-1;
      size = 5;
          arr = new int[size];
    }

   ~stack ()
    { cout << "\n destructor";
        delete [] arr;
    }
    stack (int _size)
     {  size = _size;
        tos = -1 ;
        arr = new int [size];
     }
```

S1 = S2 ⇒ donot produce errors ⇒ but tos = tos which is wrong

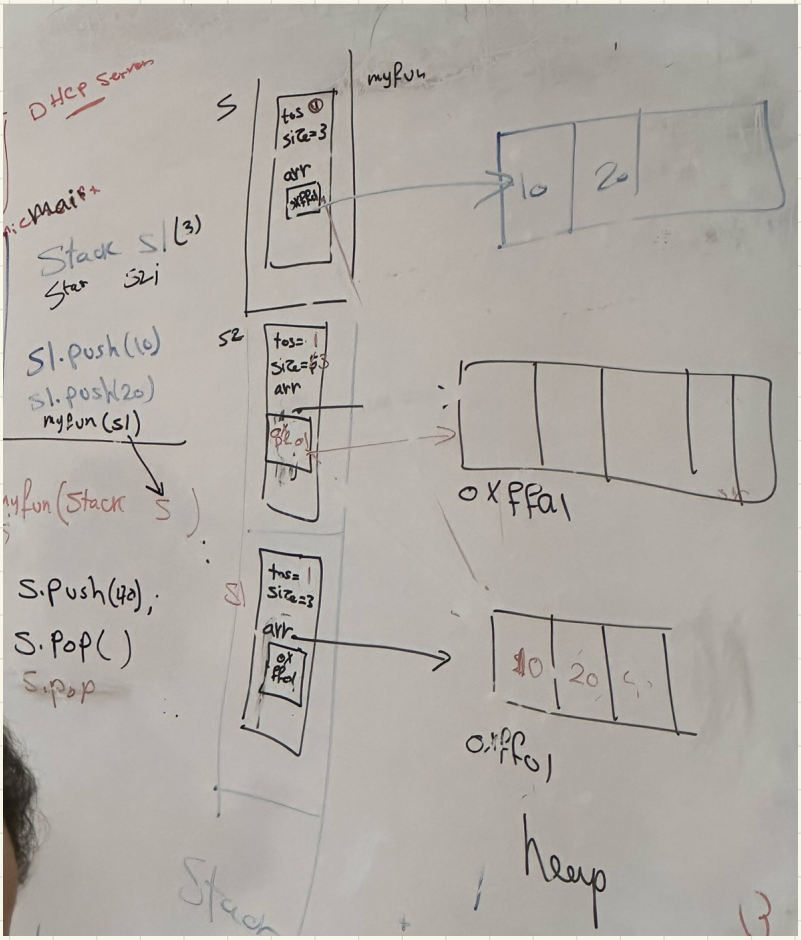
I need to define the copy constructor if I have any dynamic allocated data.

Because if I passed the data to function by value (shallow copy) it will delete the memory for the dynamic allocated data after exiting the function so I can't use it in main function

The solution is:- difining the copy constructor by myself ★

```
            cout<<"\n stack is empty";
            return 0;
        }
    }
    Stack(Stack& s){
        tos=s.tos;
        size=s.size;
        arr=new int[size];
        for(int i=0;i<=tos;i++)
            arr[i]=s.arr[i];
    }
    ~Stack(){
```

After modification


DHCP Server
Stack S1(3)
S1.push(10)
S1.push(20)
myfun(s1)
myfun(Stack S)
S.push(40);
S.Pop()
S.pop
S
myfun
tos 1
size=3
arr
10
20
S2
tos=1
size=3
arr
0xffa1
S1
tos=1
size=3
arr
0xffa1
10
20
0xffa1
Stack
Heap

Example of copy constructor

```
myfun(Stack s)
{
    int x;
    s.pop(x);
    cout<<endl<<x;
    s.pop(x);
        cout<<endl<<x;
s.Push(30);
s.Push(40);
}
int main()
{
    Stack s2(2);
    s2.Push(1);
    s2.Push(2);
    myfun(s2);
    int x;
    s2.pop(x);
    cout<<endl<<x;
```

it's undetermined behaviour so runtime error

```
2
1
 destructor
2
 destructor
Process returned 0 (0x0)   execution time : 0.230 s
Press any key to continue.
```

if I didn't create a copy constructor it will be shallow copy so reffer to same array and delete it after finishing.

LAB

* stack static

// dynamic

random notes

taken in my note book

```
Complex *ptr = new Complex
();
```

NOTE:-

1. ⇒ new creates an object and point at it using the pointer.

1

```
void myfun (Complex C)
{ C.print(); }

int main()
{ Complex C1(10,20);
  myfun(C1);
  return 0
}
```

```
void myfun (Complex &c)
{ c.print(); }
  c.setComplex (80,70);
}
```

the only change

* alias is deleted when exit function

```
int main()
{ Complex C1(20,30);
  myfun (C1);
  C1.print(); return 0; }
```

3

```
Complex C2(10,50);
Complex & ptr = C2;
ptr.print();
```

Refrence

```
Complex *ptr = new Complex(10,20)
ptr -> print();
delete (ptr);
```

Heap

```
complex & ptr = C1;
```

✓

* Complex &ptr → Compilation error

Alias

2

Copy Constractor

* Creation in Class :-

```
Complex (Complex& C3)
{
  real = C3.real;
  img = C3.img;
  Cout << "Copy Const";
}
```

alias not Creating new obj.

5

In alias example :- (Copy Constructor) if object is sent to function by value a copy constructor created by default is copying the data from the main to function using bit wise operation (bit wise copy)

In the function it's working with a copy if by value and destroying the object after that during exiting the function

4

the C3 is Same as C1

So if

Cout << "the address of"

6