OOP Course

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

Lecture 2

struct in c & c++ ⇒ functions & access modifier (default public)

class ⇒ default private ⇒ till now its the only diffrence

void print (emp e) ⇒ called stand alone function ⇒ can be called directly

{ cout << e.getId() << ":" << e.getName() ;

e.setId (9000) ; ⇒ call by value

}

functions must be after class &struct & include

- instant member or function member ⇒ functions of classes

```
void print(emp e);

int main()
{
    emp e1;
    e1.setId(10);
    e1.setName("aly");
    e1.setAge(30);
    print2(&e1);
    print(e1);   ⇒ creates a copy in function
    return 0;
}
//stand-alone function
void print(emp e)//
{
    cout<<e.getId()<<":"<<e.getName()<<":"<<e.getAge();
    e.setId(9000);   ⇒ will be deleted after exiting the function because it's a copy
}
void print2(emp* e)
{
    cout<<e->getId()<<":"<<e->getName()<<":"<<e->getAge();
    e -> setId (9000);   ⇒ will save it even after exiting function
```

* any member function has a hidden parameter called this pointing to its object

* if you debug the code you will find this

* this -> id because this a pointer

```
class Complex
{
    int real;
    int img;

    Public :
        //setters & getters   void

        void setReal ( int _real)
            { real = _real ; }


        void setimg (int _img)
            { img = _img ; }

        int getReal ()
        { return real ; }

        int getimg ()
            {return img ; }

        void print ()
        {
            if (img >0)

            cout << real << "+" << img << "j" ;

            else if (img <0)

                cout << real << img << "j" ;

                else

                cout << real ;
        }

        void setComplex (int _real , int _img)

            { real = _real ;
              img = _img ;
            }
```

```
Complex Add (Complex c)
{
    Complex res;
    res.real = c.real + real;
    res.img = c.img + img;
    return res;

    // if I typed img = 9000; it will assign this value to c2
}
};

int main()
{ complex c1, c2, c3;
  c3 = c2.Add(c1);
```

by default it's this → res

Calling by value

Calling by address

in function its called by this

}

* Note :-

polymorphisum ⇒ differ different functions by number of arguments or type of them (overloading)

* over loading depends on arguments not the return type.

* Constructor is called directly after the object creation

it is used if we need to do something right after the object created

* Constructor [1] is called once

[2] has same name as the class

[3] no return type

```
emp () {  age = 30;
          id = 0;
          strcpy(name, "no name");
```

* Constructor is called parameterless Constructor because it don't have parameters to get emp()

* we can overload the Constructor.

* emp e1, e2; → will call the parameterless Constructor

emp e3 (10, "aly", 30); → emp (int _id , char _name[], int _age)

```
{ id = _id;
  age = _age;
  strcpy (name, _name);
}
```

- if I typed just emp e1, e2; → No error
- if I typed emp e3 (10, "aly", 30); → with

```
38      }
39      void print(){
40       cout<<endl<<id<<":"<<name<<":"<<age;
41      }
42
43      emp(int _id,char _name[]){
44       id=_id;
45       strcpy(name,_name);
46       age=30;
47      }
48      emp(int _id,char _name[],int _age){
49       id=_id;
50       strcpy(name,_name);
51       age=_age;
52      }
53  };
54  void print(emp e);
55  void print2(emp* e);
56
57  int main()
58  {
59      emp e1,e2;
```

* Constructor can be private

* when you create any constructors the default const. needs to be created too or error if you call it or emp e1;

* destructor is ~name_of_class

* destructor is called when object is deleted or code is finished

* if the function the objects created in is not ended the object can't be deleted because it's saved in stack

```
int main()
{ emp* e1;        ⇒ not object
  emp * e2 = new emp (10, "ahmed");
  e2 → print();
  delete e2;
}
```

calling instructor

I have created pointer that created an object (place to save the object)

delete e2; ⟹ to delete it from the heap it will call destructor to delete its object

* Copy Constructor

used in 4 cases

#search

* Creating object from object call the copy constructor.

Exam :- → mcq.

→ 2 questions code typing.

LAB

1. Complex → setter,
getter
print
constractor
Add (member function & stand alone)

2. emp class → Constructor
destractor
trace