Day 8

13 April 2024 08:25 AM

Constructor & Destructor

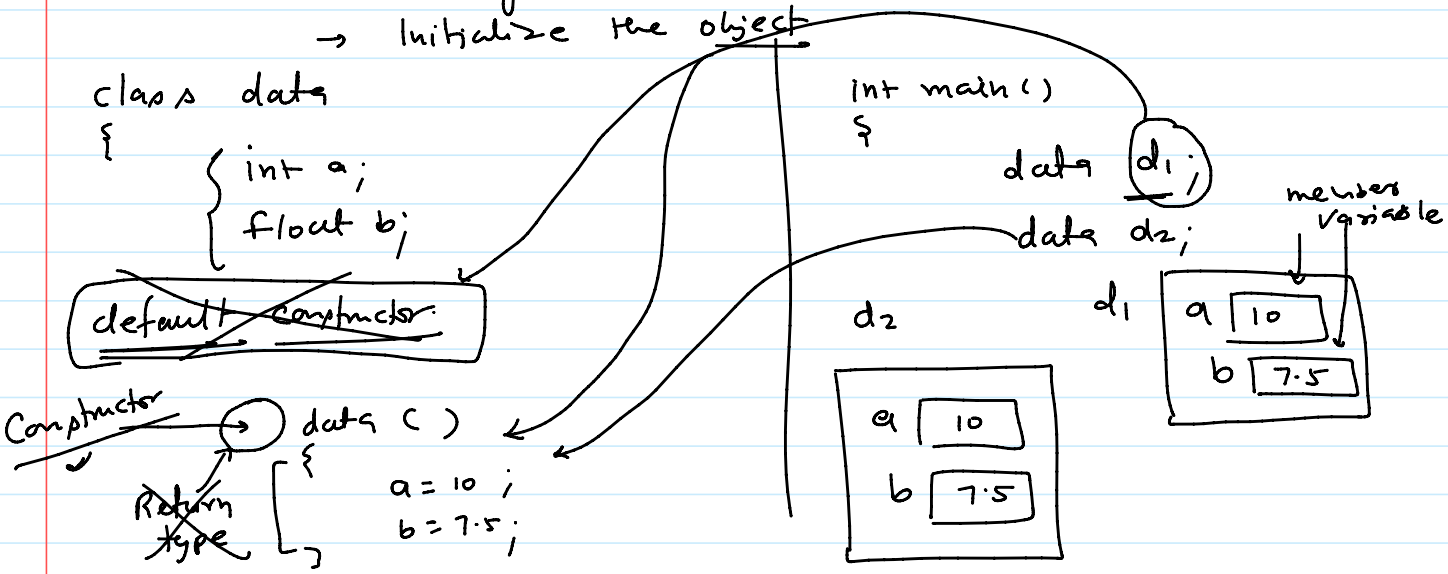
Constructor → member function

↑

name → same as class name

use → memory Allocation of member variable

→ Initialize the object



.

```
#include<iostream>
using namespace std;
class data
{
    int x;
    float y;
    public:
        data()
        {
            x=10;
            y=8.5;
            cout<<"Constructor called\n";
        }
        void output()
        {
            cout<<x<<' '<<y<<endl;
        }
};
int main()
{
    data d1;
    d1.output();
    data d2;
    d2.output();
}
```

Type of Constructor :→

✓ ① Default Constructor (zero parameters)

→ ② Parameterized Constructor
- (A) Argument ✓
- (B) Default Argument ✓
- (C) Dynamic Constructor ✓

- ③ Copy Constructor

same name & diff Argument → overloading

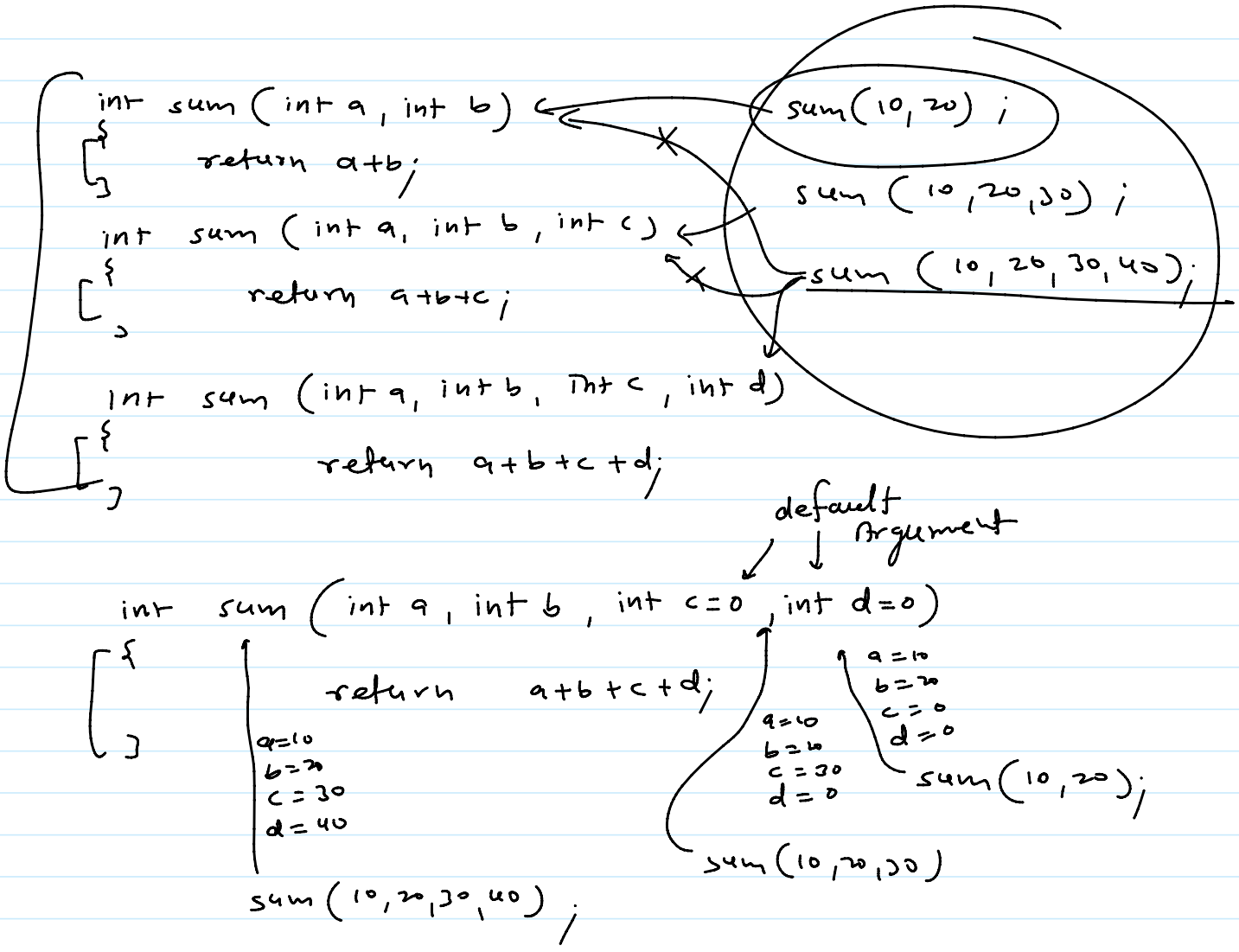
10.5f → float
10.5 → double

```
#include<iostream>
using namespace std;
class data
{
    int x;
    float y;
    public:
        data()      //default constructor
        {
            x=10;
            y=8.5;
            cout<<"Constructor called\n";
        }
        data(int x) //one parameterized
constructor
        {
            this->x=x;
            y=0.0;
            cout<<"int one parameterized
called\n";
        }
        data(float y)   //one
parameterized constructor
        {
            this->y=y;
```

```
            x=0;
            cout<<"Float one
parameterized called\n";
        }
        data(int x, float y)    //two
parameterized constructor
        {
            this->x=x;
            this->y=y;
            cout<<"two parameterized
called\n";
        }
        void output()
        {
            cout<<x<<' '<<y<<endl;
        }
};
int main()
{
    data d1;              //called default
    d1.output();
    data d2=10;     //one parameterized
    data d3(10);    //one parameterized
    data d4=data(10.5f);   //one
parameterized
    d2.output();
    data d5(10,7.5f);       //2
parameterized
    data d6=data(10,7.5f);

}
```

default Argument parameterized Constructor

```
int sum (int a, int b, int c=0, int d=0)
{
    return a+b+c+d;
}
```

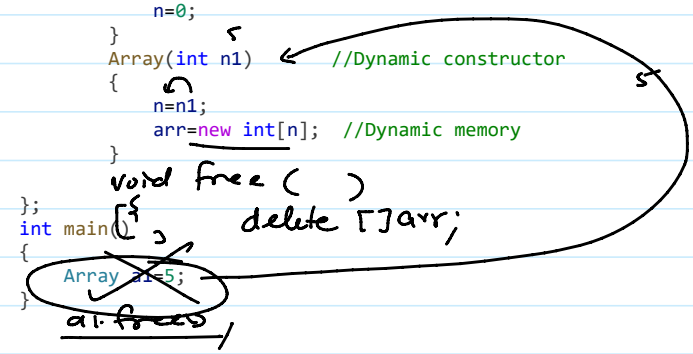


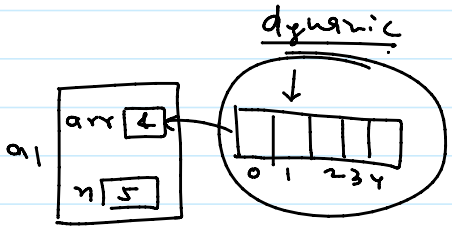

```cpp
#include<iostream>
using namespace std;
class data
{
    int x;
    float y;
    public:
        data(int x=0, float y=0.0f)
        {
            this->x=x;
            this->y=y;
            cout<<"Default argument constructor\n";
        }
        void output()
        {
            cout<<x<<' '<<y<<endl;
        }
};
int main()
{
    data d1;            //called default
    data d2=10;     //one parameterized
    data d3(10);    //one parameterized
    data d4=data(10.5f);   //one parameterized
    data d5(10,7.5f);       //2 parameterized
    data d6=data(10,7.5f);
    d1.output();
    d2.output();
    d3.output();
    d4.output();
```

```
        d5.output();
        d6.output();
    }
```

dynamic Constructor :-

```
#include<iostream>
using namespace std;
class Array
{
    int *arr;
    int n;
    public:
        Array()
        {
            arr=nullptr;
            n=0;
        }
        Array(int n1)    //Dynamic constructor
        {
            n=n1;
            arr=new int[n];  //Dynamic memory
        }
};
int main()
{
    Array a1=5;
}
```






```
#include<iostream>
using namespace std;
class Array
{
    int *arr;
    int n;
    public:
        Array()
        {
            arr=nullptr;
            n=0;
        }
        Array(int n1)       //Dynamic constructor
        {
            n=n1;
            arr=new int[n];  //Dynamic memory
        }
        void input()
        {
            cout<<"Enter "<<n<<" values:";
            for(int i=0;i<n;i++)
            {
                cin>>arr[i];
            }
        }
        void output()
        {
            for(int i=0;i<n;i++)
            {
                cout<<arr[i]<<" ";
            }
            cout<<endl;
        }
        void free()
        {
            delete []arr;
```

```
        }

};
int main()
{
    int n;
    cout<<"Enter no of element:";
    cin>>n;
    Array a1=n;
    a1.input();
    a1.output();
    a1.free();
    return 0;
}
```

1.Constructors is a member function.
2.It is called automatically when we create an object.
3.It name is same as the class name.
4.It should be defined in public block but it can be defined as private if called inside a member function.
5.A class can contain more than one constructors if there arguments are different and this is called as constructor overloading.
6.Every class contains a default zero argument constructor if no explicit constructor is defined by us (copy constructor).
7. It has no return type.

Use of constructor:-
a.constructor is used to provide memory to class member variable inside the object .
b.It can also be used to initialize member variables.
c.constructor is not void it returns object type itself.

destructor :—

~ class name
— No argument → No overloading
— auto called

```
#include<iostream>
using namespace std;
class Array
{
    int *arr;
    int n;
    public:
        Array()
        {
            arr=nullptr;
            n=0;
        }
        Array(int n1)       //Dynamic constructor
        {
            n=n1;
            arr=new int[n];  //Dynamic memory
        }
        void input()
        {
            cout<<"Enter "<<n<<" values:";
            for(int i=0;i<n;i++)
            {
                cin>>arr[i];
            }
        }
        void output()
        {
```

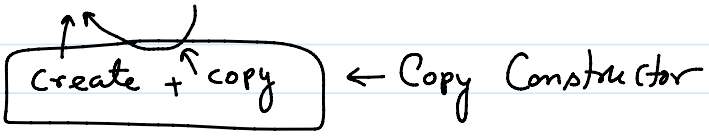
```
                for(int i=0;i<n;i++)
                {
                    cout<<arr[i]<<" ";
                }
                cout<<endl;
            }
            ~Array()
            {
                delete []arr;
            }

    };
    int main()
    {
        int n;
        cout<<"Enter no of element:";
        cin>>n;
        Array a1=n;
        a1.input();
        a1.output();
        return 0;
    }
```

```
#include<iostream>
using namespace std;
class data
{
    int a;
    public:
        data(int x)
        {
            a=x;
            cout<<a<<" Constructor called\n";
        }
        ~data()
        {
            cout<<a<<" Destructor called\n";
        }
};
int main()
{
    data d1=10;
    data d2=20;
    return 0;
}
```

Copy Constructor :→

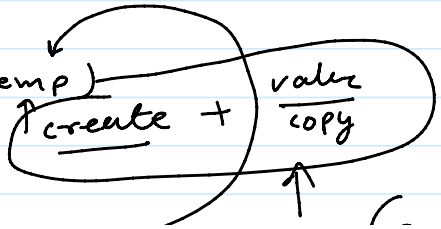
Array a1 = 5;

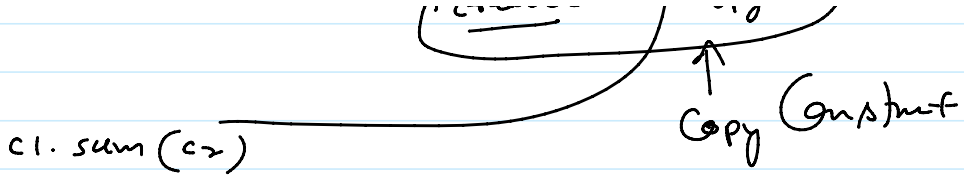
Array $a_2 = a_1$

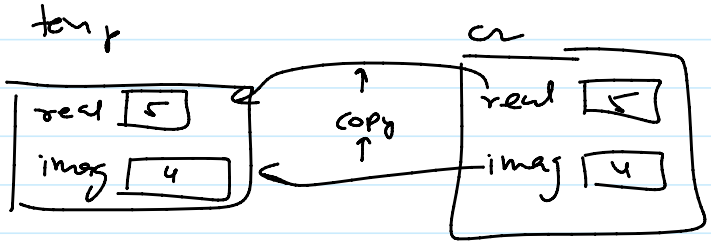


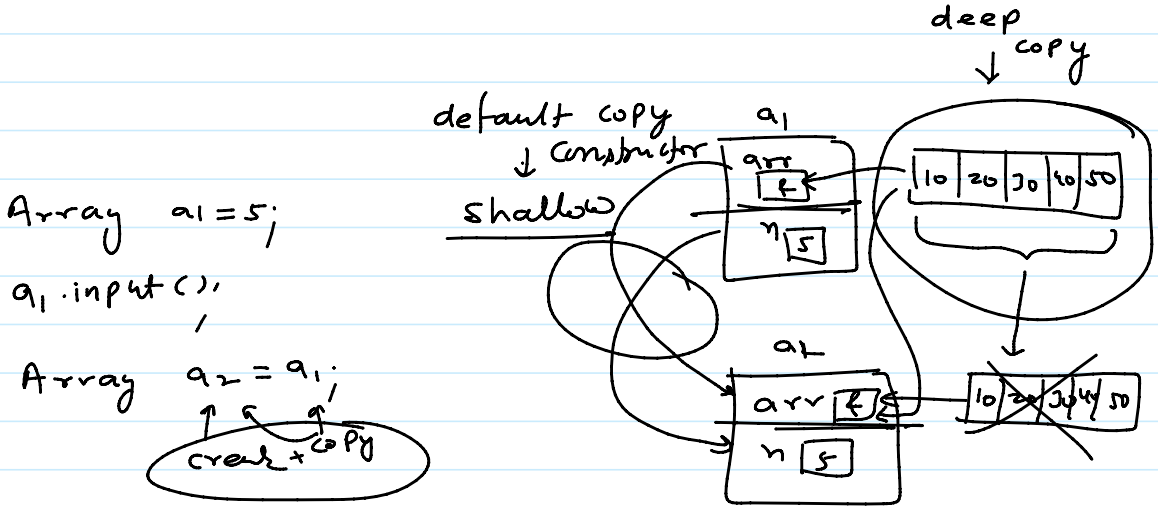
fune call → call by Value

sum (Complex temp)










```cpp
#include<iostream>
using namespace std;
class Array
{
    int *arr;
    int n;
    public:
        Array()
        {
            arr=nullptr;
            n=0;
            cout<<"Default constructor\n";
        }
        Array(int n1)       //Dynamic constructor
        {
            n=n1;
            arr=new int[n];  //Dynamic memory
            cout<<"Parameterized constructor\n";
        }
        void input()
        {
            cout<<"Enter "<<n<<" values:";
            for(int i=0;i<n;i++)
            {
                cin>>arr[i];
            }
        }
        void output()
        {
            for(int i=0;i<n;i++)
            {
                cout<<arr[i]<<" ";
            }
```

```
            cout<<endl;
        }
        ~Array()
        {
            delete []arr;
        }
        // Array(const Array &r)      //shallow copy
        // {
        //     arr=r.arr;
        //     n=r.n;
        //     cout<<"Copy constructor called\n";
        // }
        Array(const Array &r)   //deep copy
        {
            n=r.n;
            arr=new int[n];
            for(int i=0;i<n;i++)
                arr[i]=r.arr[i];
        }
};
int main()
{
    int n;
    cout<<"Enter no of element:";
    cin>>n;
    Array a1=n;
    a1.input();     //10 20 30 40 50
    Array a2=a1;
    a1.output();       //10 20 30 40 50
    a2.output();        //10 20 30 40 50
    a2.input();         //100 200 300 400 500
    a1.output();        //10 20 30 40 50
    a2.output();        //100 200 300 400 500
    return 0;
}
```

Static & Constant :→

Types of member Variable

1. Instance member Variable → (object member)
2. Static member variable (class member)

```
class data
{   public:
        int a;      ← Instance member Variable
        static int b;   ← static member Variable
};
int data::b;
```

```
int main()
{   data d1, d2;
    d1.a=10;
    d1.b=20;
    d2.a=100;
    d2.b=200;
    data::b=500;
```

d1: a [10]

d2: a [100]

b: ~~20~~ ~~200~~ 500

Error ← `data::a=500;`

Print 1 to 100

```cpp
#include<iostream>
using namespace std;
class A
{
    static int x;
    public:
        A()
        {
            x++;
            cout<<x<<" ";
        }
};
int A::x=0;
int main()
{
    A obj[100];
}
```

member function :-

1. Instance
2. static

```
class data
{
        int x;
        static int y;
        public :
                void fun1 ()
                {
                  this->x=10;
                      y=20;
                }
                static void fun2 ()
                {
                      x=100;
                      y=200;
                }
};
int data::y;

int main ()
{
        data d1;
        d1.fun1 ();
        d1.fun2 ();
        data::fun2 ();
```

Instance member function

this

static member function

Error