

C, C++, DSA in depth

Initializer



Saurabh Shukla (MySirG)

Agenda

- ① Initialization list
- ② Initializing fields
- ③ Initializing const members
- ④ Initializing reference variables

Initializers

Also known as initializer list.

It allows you which constructor is called and what arguments that constructor receives.

If you have a reference or a const field or if one of the classes used does not have a default constructor, you must use an initialization list.

```
class ClassName
```

```
{ public:
```



```
    ClassName(args): initialization list  
    {
```

```
    }
```

```
}
```

Using initialization list to initialize fields

class A

{

private:

int a, b, c;

public:

A() : a(5), b(6), c(8)

{ }

A(int a, int b, int c) : a(a), b(b), c(c)

{ }

};

fields

Resolving name conflict

Initializing Const member

```
class A  
{
```

```
    private:
```

```
        const int K;
```

```
    public:
```

```
        A()  
        {
```

```
            K=10;
```

```
        }
```

ERROR

```
        A(): K(10)  
        {
```

```
            }
```

```
        }
```

CORRECT

```
};
```

const member variable
can only be initialized using
initializer list.

You cannot modify const
member variable.

Initializing reference variable

```
class A  
{
```

```
    private:
```

```
        int &x;
```

```
        int k;
```

```
    public:
```

```
        A() : k(5), x(k)
```

```
        { }
```

```
};
```

Reference variable



$x(*\text{new int})$



Where you can use initializer list

- ① To initialize reference member variable
- ② To initialize const member variable
- ③ To initialize data members (fields)
- ④ To invoke base class constructor