

constructor with inheritance Multiple inheritance Hybrid one

1. Constructor with Inheritance

when a derived class object is created, the **base class constructor is called first**, followed by the **derived class constructor**.

```
#include <iostream>
```

```
using namespace std;
```

```
class Base {
```

```
public:
```

```
    Base()
```

```
    {
```

```
        cout << "Base constructor with inheritance called" << endl;
```

```
    }
```

```
};
```

```
class Derived : public Base
```

```
{
```

```
public:
```

```
    Derived()
```

```
    {
```

```
        cout << "Derived constructor with inheritance called" << endl;
```

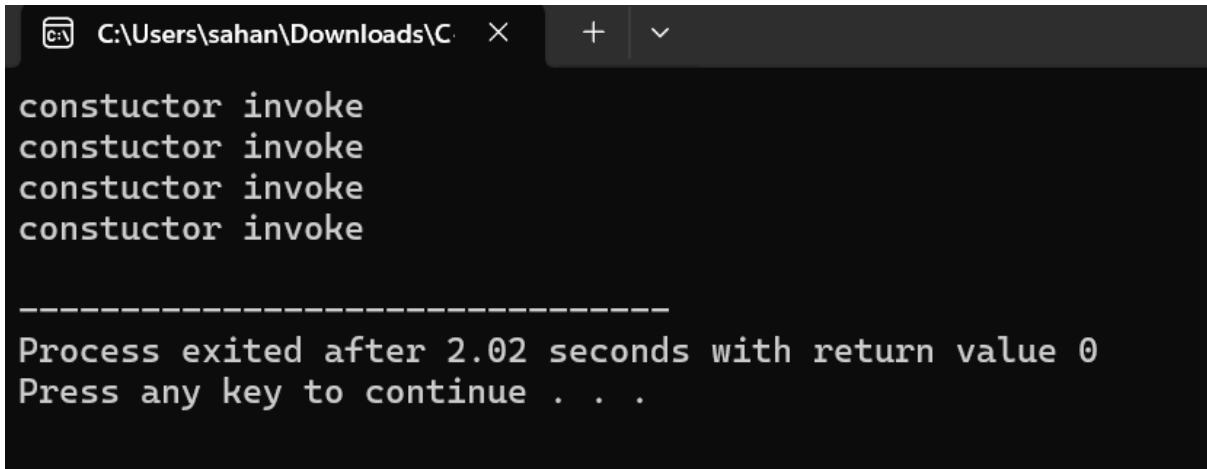
```
    }
```

```
};
```

```
int main() {
```

```
    Derived obj;
```

```
    return 0;
}
```



```
C:\Users\sahan\Downloads\C... X + v
constuctor invoke
constuctor invoke
constuctor invoke
constuctor invoke

-----
Process exited after 2.02 seconds with return value 0
Press any key to continue . . .
```

2. Multiple Inheritance with Constructor

```
#include <iostream>

using namespace std;

class A {
public:
    A() {
        cout << "Constructor of A" << endl;
    }
};

class B {
public:
    B() {
        cout << "Constructor of B" << endl;
    }
};

class C : public A, public B {
public:
    C() {
```

```

        cout << "Constructor of C" << endl;

    });

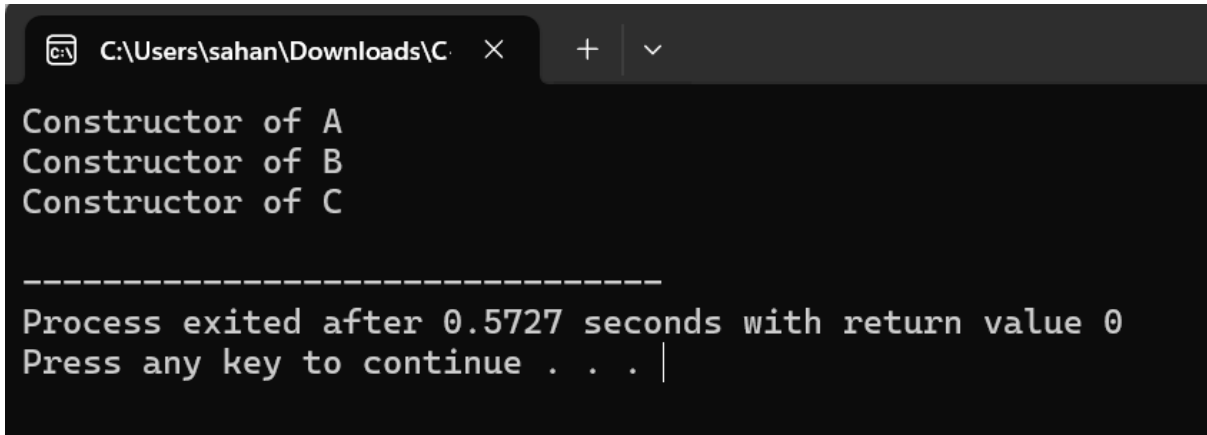
int main() {

    C obj;

    return 0;

}

```



```

C:\Users\sahan\Downloads\C >
Constructor of A
Constructor of B
Constructor of C

-----
Process exited after 0.5727 seconds with return value 0
Press any key to continue . . . |

```

3. Hybrid Inheritance with Constructor

Hybrid = Combination of multiple + hierarchical inheritance.

```
#include <iostream>
```

```
using namespace std;
```

```

class A {
public:
    A() {
        cout << "A Constructor" << endl;
    }
};

```

```

class B : public A {
public:
    B() {

```

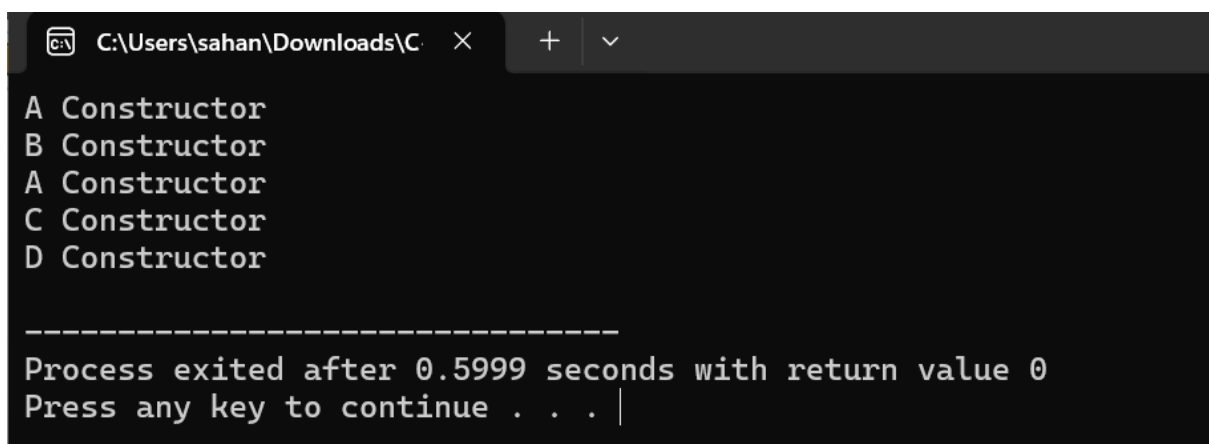
```
        cout << "B Constructor" << endl;
    }
};

class C : public A {
public:
    C() {
        cout << "C Constructor" << endl;
    }
};

class D : public B, public C {
public:
    D() {
        cout << "D Constructor" << endl;
    }
};

int main() {
    D obj;

    return 0;
}
```



The screenshot shows a Windows file explorer window with the file path `C:\Users\sahan\Downloads\C...`. Below the window is a black terminal window with white text. The terminal output shows the sequence of constructor calls: `A Constructor`, `B Constructor`, `A Constructor`, `C Constructor`, and `D Constructor`. A horizontal line separates this from the final output: `Process exited after 0.5999 seconds with return value 0` and `Press any key to continue . . . |`.

```
C:\Users\sahan\Downloads\C... × + v
A Constructor
B Constructor
A Constructor
C Constructor
D Constructor

-----
Process exited after 0.5999 seconds with return value 0
Press any key to continue . . . |
```