```cpp
// ----------Protected member -----------

#include <iostream>
using namespace std;

class baseclass
{
    private:
        int id_private {};

    public:
        int id_public {};

    protected:
        int id_protected {};

};


class derivedclass : public baseclass
{
    public:
        void setId(int id)
        {
            id_public = id+2;
            id_protected = id;
        }

        void displayId()
        {
            cout << "id_protected is: " << id_protected << endl;
            cout << "id_public is: " << id_public << endl;
        }
};

int main() {

    derivedclass obj1;
    obj1.setId(81);
    obj1.displayId();
    return 0;
}
```

Output:

```
id_protected is: 81

id_public is: 83
```

```cpp
// ----------Single Inheritance -----------

#include <iostream>
using namespace std;

class baseclass
{
    public:

    baseclass(){
        cout << "base class constructor" << endl;
    }
    ~baseclass(){
        cout << "base class destructor" << endl;
    }

};

class derivedclass : public baseclass
{

    public:

    derivedclass(){
        cout << "derived class constructor" << endl;
    }
    ~derivedclass(){
        cout << "derived class destructor" << endl;
    }
};

int main() {

    derivedclass obj1;

    return 0;
}
```

Output:

```
base class constructor

derived class constructor

derived class destructor

base class destructor
```

```cpp
// ----------multiple Inheritance -----------

#include <iostream>
using namespace std;

class baseclass_a
{
    public:
    int x{};
    baseclass_a(){
        cout << "baseclass_a constructor" << endl;
    }
    ~baseclass_a(){
        cout << "baseclass_a destructor" << endl;
    }

};

class baseclass_b
{
    public:
    int y {};
    baseclass_b(){
        cout << "baseclass_b constructor" << endl;
    }
    ~baseclass_b(){
        cout << "baseclass_b destructor" << endl;
    }
};

class derivedclass : public baseclass_a , public baseclass_b
{

    public:
    int z {};
    derivedclass(){
        cout << "derived class constructor" << endl;
    }
    ~derivedclass(){
        cout << "derived class destructor" << endl;
    }
};

int main() {

    derivedclass obj1;
    obj1.x = 56;
    obj1.y = 85;
    obj1.z = 98;

    return 0;
}
```

Output:

```
baseclass_a constructor

baseclass_b constructor

derived class constructor

derived class destructor

baseclass_b destructor

baseclass_a destructor
```

```cpp
// ----------Multi level Inheritance & Constructor and destructors-----------

#include <iostream>
using namespace std;


class baseclass
{
    public:
    int x{};
    baseclass(){
        cout << "baseclass constructor" << endl;
    }
    ~baseclass(){
        cout << "baseclass destructor" << endl;
    }

};

class derivedclass_a: public baseclass
{
    public:
    int y {};
    derivedclass_a(){
        cout << "derivedclass_a constructor" << endl;
    }
    ~derivedclass_a(){
        cout << "derivedclass_a destructor" << endl;
    }

};

class derivedclass_b : public derivedclass_a
{

    public:
    int z {};
    derivedclass_b(){
        cout << "derivedclass_b constructor" << endl;
    }
    ~derivedclass_b(){
        cout << "derivedclass_b destructor" << endl;
    }
};

int main() {

    derivedclass_b obj1;
    obj1.x = 56;
    obj1.y = 85;
    obj1.z = 98;

    return 0;
}
```

Result:

```
baseclass constructor

derivedclass_a constructor

derivedclass_b constructor

derivedclass_b destructor

derivedclass_a destructor

baseclass destructor
```

```cpp
// ----------passing args to base class constructors -----------

#include <iostream>
using namespace std;

class baseclass
{
    public:
    int value{};
    baseclass():value{25}{
        cout << "baseclass constructor without args" << endl;
    }
    baseclass(int x):value{x}{
        cout << "baseclass constructor with args" << endl;
    }
    ~baseclass(){
        cout << "baseclass destructor" << endl;
    }

};


class derivedclass : public baseclass
{

    public:
    int der_value {};
    derivedclass():baseclass{50}, der_value {30}{
        cout << "derived class constructor with-out args" << endl;
    }
    derivedclass(int x):baseclass{10*x}, der_value {x}{
        cout << "derived class constructor with args" << endl;
    }
    ~derivedclass(){
        cout << "derived class destructor" << endl;
    }
};

int main() {

    derivedclass obj1;
    cout << "der_val:" << obj1.der_value << " & base value:" << obj1.value <<
endl;

    return 0;
}
```

Output:

```
baseclass constructor with args

derived class constructor with-out args

der_val:30 & base value:50

derived class destructor

baseclass destructor
```