// ----------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

// ----------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

// ----------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

// ----------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

// ----------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