Contents

[Polymorphism sample programs 1](#_Toc96417830)

[Static Linking 1](#_Toc96417831)

[Program: 1](#_Toc96417832)

[Result: 2](#_Toc96417833)

[static binding with pointers 3](#_Toc96417834)

[program: 3](#_Toc96417835)

[Result: 4](#_Toc96417836)

[dynamic binding  eg1 5](#_Toc96417837)

[program: 5](#_Toc96417838)

[Result: 6](#_Toc96417839)

[dynambic binding eg2  constructors 7](#_Toc96417840)

[program: 7](#_Toc96417841)

[Result: 8](#_Toc96417842)

[dynambic binding eg2  constructors & destructors 9](#_Toc96417843)

[program: 9](#_Toc96417844)

[Result: 10](#_Toc96417845)

# Polymorphism sample programs

## Static Linking

### Program:

// -------------------------- static linking ----------------------------------

#include <iostream>

using namespace std;

class base\_account{

    public:

        base\_account(){

            cout << "base class constructor" << endl;

        }

        void func(){

            cout << "base\_account class function" << endl;

        }

};

class derived\_account:public base\_account{

    public:

        derived\_account(){

            cout << "derived class constructor" << endl;

        }

        void func(){

            cout << "derived\_account class function" << endl;

        }

};

void global\_func(base\_account obj){

    cout<< "global function" << endl;

    obj.func();

}

int main()

{

    cout << "========================" << endl;

    base\_account base\_obj;

    base\_obj.func();

    global\_func(base\_obj);

    cout << "========================" << endl;

    derived\_account derived\_obj;

    derived\_obj.func();

    global\_func(derived\_obj);

cout << "========================" << endl;

    return 0;

}

### Result:

========================

base class constructor

base\_account class function

global function

base\_account class function

========================

base class constructor

derived class constructor

derived\_account class function

global function

base\_account class function

========================

## static binding with pointers

### program:

// ---static binding with pointers -----

#include <iostream>

using namespace std;

class base\_account{

    public:

        base\_account(){

            cout << "base class constructor" << endl;

        }

        void func(){

            cout << "base\_account class function" << endl;

        }

};

class derived\_account:public base\_account{

    public:

        derived\_account(){

            cout << "derived class constructor" << endl;

        }

        void func(){

            cout << "derived\_account class function" << endl;

        }

};

void global\_func(base\_account obj){

    cout<< "global function" << endl;

    obj.func();

}

int main()

{

    cout << "========================" << endl;

    base\_account base\_obj;

    base\_obj.func();

    global\_func(base\_obj);

    cout << "========================" << endl;

    derived\_account derived\_obj;

    derived\_obj.func();

    global\_func(derived\_obj);

    cout << "========================" << endl;

    base\_account \*ptr = new derived\_account();

    ptr->func();

    cout << "========================" << endl;

    return 0;

}

### Result:

========================

base class constructor

base\_account class function

global function

base\_account class function

========================

base class constructor

derived class constructor

derived\_account class function

global function

base\_account class function

========================

base class constructor

derived class constructor

base\_account class function

========================

## dynamic binding  eg1

### program:

//-------- dynamic binding  eg1 -----------

#include <iostream>

using namespace std;

class base\_account{

    public:

        base\_account(){

            cout << "base class constructor" << endl;

        }

        virtual void func() const{

            cout << "base\_account class function" << endl;

        }

};

class derived\_account:public base\_account{

    public:

        derived\_account(){

            cout << "derived class constructor" << endl;

        }

        void func() const{

            cout << "derived\_account class function" << endl;

        }

};

void global\_func(const base\_account &obj){

    cout<< "global function" << endl;

    obj.func();

}

int main()

{

    cout << "================" << endl;

    base\_account base\_obj;

    //base\_obj.func();

    global\_func(base\_obj);

    cout << "================" << endl;

    derived\_account derived\_obj;

    derived\_obj.func();

    global\_func(derived\_obj);

    cout << "================" << endl;

    base\_account \*ptr = new derived\_account();

    ptr->func();

    cout << "================" << endl;

    return 0;

}

### Result:

================

base class constructor

global function

base\_account class function

================

base class constructor

derived class constructor

derived\_account class function

global function

derived\_account class function

================

base class constructor

derived class constructor

derived\_account class function

================

## dynambic binding eg2  constructors

### program:

// ------- dynambic binding eg2  constructors---------

#include <iostream>

using namespace std;

class base\_account{

    public:

        base\_account(){

            cout << "base class constructor" << endl;

        }

        virtual void func() const{

            cout << "base\_account class function" << endl;

        }

};

class derived\_account\_a:public base\_account{

    public:

        derived\_account\_a(){

            cout << "derived\_a class constructor" << endl;

        }

        virtual void func() const{

            cout << "derived\_account\_a class function" << endl;

        }

};

class derived\_account\_b:public base\_account{

    public:

        derived\_account\_b(){

            cout << "derived\_b class constructor" << endl;

        }

        virtual void func() const{

            cout << "derived\_account\_b class function" << endl;

        }

};

void global\_func(const base\_account &obj){

    cout<< "global function" << endl;

    obj.func();

}

int main()

{

    cout << "================" << endl;

    base\_account base\_obj;

    //base\_obj.func();

    global\_func(base\_obj);

    cout << "================" << endl;

    derived\_account\_a derived\_obj;

    //derived\_obj.func();

    global\_func(derived\_obj);

    cout << "================" << endl;

    base\_account \*ptr1 = new base\_account();

    base\_account \*ptr2 = new derived\_account\_a();

    base\_account \*ptr3 = new derived\_account\_b();

    ptr1->func();

    ptr2->func();

    ptr3->func();

    cout << "================" << endl;

    delete ptr1;

    delete ptr2;

    delete ptr3;

    cout << "================" << endl;

    return 0;

}

### Result:

================

base class constructor

global function

base\_account class function

================

base class constructor

derived\_a class constructor

global function

derived\_account\_a class function

================

base class constructor

base class constructor

derived\_a class constructor

base class constructor

derived\_b class constructor

base\_account class function

derived\_account\_a class function

derived\_account\_b class function

================

================

## dynambic binding eg2  constructors & destructors

### program:

// ------- dynambic binding eg2  constructors & destructors --------

#include <iostream>

using namespace std;

class base\_account{

    public:

        base\_account(){

            cout << "base class constructor" << endl;

        }

        virtual void func() const{

            cout << "base\_account class function" << endl;

        }

        virtual ~base\_account(){

            cout << "base class destructor" << endl;

        }

};

class derived\_account\_a:public base\_account{

    public:

        derived\_account\_a(){

            cout << "derived\_a class constructor" << endl;

        }

        virtual void func() const{

            cout << "derived\_account\_a class function" << endl;

        }

        virtual ~derived\_account\_a(){

            cout << "derived\_account\_a destructor" << endl;

        }

};

class derived\_account\_b:public base\_account{

    public:

        derived\_account\_b(){

            cout << "derived\_b class constructor" << endl;

        }

        virtual void func() const{

            cout << "derived\_account\_b class function" << endl;

        }

        virtual ~derived\_account\_b(){

            cout << "derived\_account\_b destructor" << endl;

        }

};

void global\_func(const base\_account &obj){

    cout<< "global function" << endl;

    obj.func();

}

int main()

{

    cout << "================" << endl;

    base\_account \*ptr1 = new base\_account();

    base\_account \*ptr2 = new derived\_account\_a();

    base\_account \*ptr3 = new derived\_account\_b();

    ptr1->func();

    ptr2->func();

    ptr3->func();

    cout << "================" << endl;

    delete ptr1;

    delete ptr2;

    delete ptr3;

    cout << "================" << endl;

    return 0;

}

### Result:

================

base class constructor

base class constructor

derived\_a class constructor

base class constructor

derived\_b class constructor

base\_account class function

derived\_account\_a class function

derived\_account\_b class function

================

base class destructor

derived\_account\_a destructor

base class destructor

derived\_account\_b destructor

base class destructor

================