**Program No:** 01

**Program Topic:** *Inheritance (Base Class Access Control)*

**Program Title:**

Write a C++ program to inherit a base class as public.

**Objectives:**

To learn how to access a base class as public by a derived class.

**Syntax:**

class derived\_class\_name : access base\_class\_name

{

// ...

}

**Source Code:**

#include <iostream>

using namespace std;

class base

{

int x;

public:

void setx(int n){ x = n; }

void showx( ){cout << x << endl;}

};

class derived : public base

{

int y;

public:

void sety(int n) { y = n; }

void showy( ){cout << y << endl;}

};

int main ( )

{

derived ob;

ob. setx (10);

ob. sety (20);

ob. showx ();

ob. showy ();

return 0;

}

**Output:**

10

20

Process returned 0 (0x0) execution time : 0.071 s

Press any key to continue.

**Program No:** 02

**Program Topic:** *Inheritance (Base Class Access Control)*

**Program Title:**

Write a C++ program to inherit a base class as private.

**Objectives:**

To learn how to access a base class as private by a derived class.

**Syntax:**

class derived\_class\_name : access base\_class\_name

{

// ...

}

**Source Code:**

#include <iostream>

using namespace std;

class base

{

int x;

public :

void setx (int n){x = n;}

void showx( ){cout << x << endl; }

};

class derived : private base

{

int y;

public:

void setxy(int n, int m){setx (n); y = m;}

void showxy( ){showx (); cout << y << endl;}

};

int main( )

{

derived ob;

ob.setxy (10 , 20) ;

ob.showxy( );

return 0;}

**Output:**

10

20

Process returned 0 (0x0) execution time : 0.054 s

Press any key to continue.

**Program No:** 03

**Program Topic:** *Inheritance (Protected Members)*

**Program Title:**

Write a C++ program using protected members of a class.

**Objectives:**

To learn about protected members.

**Syntax:**

class class\_name

{

// private members

protected : // optional

// protected members

public :

// public members

};

**Source Code:**

#include <iostream>

using namespace std;

class samp

{

int a;

protected:

int b;

public:

int c;

samp ( int n, int m) { a = n; b = m; }

int geta () { return a; }

int getb () { return b; }

};

int main ()

{

samp ob (10 , 20) ;

ob.c = 30;

cout << ob. geta () << ' ';

cout << ob. getb () << ' ' << ob.c << endl;

return 0;

}

**Output:**

10 20 30

Process returned 0 (0x0) execution time : 0.079 s

Press any key to continue.

**Program No:** 04

**Program Topic:** *Inheritance (Protected Members)*

**Program Title:**

Write a C++ program to inherit protected members as public.

**Objectives:**

To learn how to inherit protected members as public.

**Syntax:**

class class\_name

{

// private members

protected : // optional

// protected members

public :

// public members

};

**Source Code:**

#include <iostream>

using namespace std;

class base

{

protected:

int a, b;

public :

void setab ( int n, int m) { a = n; b = m; }

};

class derived : public base

{

int c;

public :

void setc(int n){c = n;}

void showbc( )

{

cout << a << ' ' << b << ' ' << c << endl;

}

};

int main( )

{

derived ob;

ob.setab(1, 2);

ob.setc(3);

ob.showbc( );

return 0;

}

**Output:**

1 2 3

Process returned 0 (0x0) execution time : 0.137 s

Press any key to continue.

**Program No:** 05

**Program Topic:** *Inheritance (Constructor & Destructors)*

**Program Title:**

Write a C++ program using constructor and destructor in a base and derived class.

**Objectives:**

To learn how to execute constructors and destructors in base and derived class and the sequence of their calls.

**Syntax:**

derived\_constructor ( arg\_list ): base ( arg\_list )

{

// body of derived class constructor

}

**Source Code:**

#include <iostream>

using namespace std;

class base

{

public :

base ( ){cout << "Constructing base class \n"; }

~ base ( ){cout << "Destructing base class \n"; }

};

class derived : public base

{

public :

derived ( ) {cout << "Constructing derived class \n"; }

~ derived ( ) {cout << "Destructing derived class \n"; }

};

int main ( )

{

derived o;

return 0;

}

**Output:**

Constructing base class

Constructing derived class

Destructing derived class

Destructing base class

Process returned 0 (0x0) execution time : 0.075 s

Press any key to continue.

**Program No:** 06

**Program Topic:** *Inheritance (Constructor & Destructors)*

**Program Title:**

Write a C++ program to pass arguments on constructor and destructor of base and derived class.

**Objectives:**

To learn how to pass arguments on constructor and destructor of base and derived class.

**Syntax:**

base ( arg\_list)

{

//body

}

… … …

derived\_constructor ( arg\_list ): base ( arg\_list )

{

// body of derived class constructor

}

**Source Code:**

#include <iostream>

using namespace std;

class base

{

int i;

public :

base (int n)

{

cout << " Constructing base class \n";

i = n;

}

~base () {cout << " Destructing base class \n"; }

void showi () { cout << i << endl; }

};

class derived : public base{

int j;

public :

derived ( int n, int m) : base (m)

{

cout << " Constructing derived class \n";

j = n;

}

~derived () { cout << " Destructing derived class \n"; }

void showj () { cout << j << endl; }

};

int main ()

{

derived o(10 , 20) ;

o. showi ();

o. showj ();

return 0;

}

**Output:**

Constructing base class

Constructing derived class

20

10

Destructing derived class

Destructing base class

Process returned 0 (0x0) execution time : 0.091 s

Press any key to continue.

**Program No:** 07

**Program Topic:** *Multiple Inheritance*

**Program Title:**

Write a C++ program of a derived class that inherits a class derived from another class.

**Objectives:**

To learn about multiple inheritance.

**Syntax:**

class derived\_class\_name : access base1 , access base2 , ... ,

access baseN

{

// ... body of class

}

derived\_constructor ( arg\_list ) : base1 ( arg\_list ), base2 ( arg\_list ),

... , baseN (arg - list )

{

// body of derived class constructor

}

**Source Code:**

# include <iostream>

using namespace std;

class B1

{

int a;

public :

B1( int x) { a = x; }

int geta () { return a; }

};

class D1 : public B1

{

int b;

public:

D1( int x, int y) : B1(y)

{

b = x;

}

int getb () { return b; }

};

class D2 : public D1

{

int c;

public:

D2( int x, int y, int z) : D1(y, z)

{

c = x;

}

void show ()

{

cout << geta () << ' ' << getb () << endl;

cout << c << endl;

}

};

int main ()

{

D2 ob (1, 2, 3);

ob. show ();

cout << ob. geta () << ' ' << ob. getb () << endl;

return 0;

}

**Output:**

3 2

1

3 2

Process returned 0 (0x0) execution time : 0.086 s

Press any key to continue.

**Program No:** 08

**Program Topic:** *Multiple Inheritance*

**Program Title:**

Write a C++ program to inherit two base classes.

**Objectives:**

To learn how to inherit two base classes.

**Syntax:**

class derived\_class\_name : access base1 , access base2 , ... ,

access baseN

{

// ... body of class

}

derived\_constructor ( arg\_list ) : base1 ( arg\_list ), base2 ( arg\_list ),

... , baseN (arg - list )

{

// body of derived class constructor

}

**Source Code:**

# include <iostream>

using namespace std;

class B1

{

int a;

public :

B1( int x) { a = x; }

int geta () { return a; }

};

class B2

{

int b;

public :

B2(int x) { b = x; }

int getb () { return b; }

};

class D : public B1 , public B2

{

int c;

public :

D( int x, int y, int z) : B1(z), B2(y)

{

c = x;

}

void show ()

{

cout << geta () << ' ' << getb () << endl;

cout << c << endl;

}

};

int main ()

{

D ob (1, 2, 3);

ob. show ();

return 0;

}

**Output:**

3 2

1

Process returned 0 (0x0) execution time : 0.051 s

Press any key to continue.

**Program No:** 09

**Program Topic:** *Multiple Inheritance (Constructors & Destructors)*

**Program Title:**

Write a C++ program that explains constructor and destructor calls for multiple base class.

**Objectives:**

To learn about the sequence of constructor and destructor calls for multiple base class.

**Syntax:**

class derived\_class\_name : access base1 , access base2 , ... ,

access baseN

{

// ... body of class

}

derived\_constructor ( arg\_list ) : base1 ( arg\_list ), base2 ( arg\_list ),

... , baseN (arg - list )

{

// body of derived class constructor

}

**Source Code:**

# include <iostream>

using namespace std;

class B1

{

public :

B1( ) { cout << " Constructing B1\n"; }

~B1( ) { cout << " Destructing B1\n"; }

};

class B2

{

public :

B2 ( ) { cout << " Constructing B2\n"; }

~B2 ( ) { cout << " Destructing B2\n"; }

};

class D : public B1 , public B2

{

public :

D( ) { cout << " Constructing D\n"; }

~D( ) { cout << " Destructing D\n"; }

};

int main ( )

{

D ob;

return 0;

}

**Output:**

Constructing B1

Constructing B2

Constructing D

Destructing D

Destructing B2

Destructing B1

Process returned 0 (0x0) execution time : 0.047 s

Press any key to continue.

**Program No:** 10

**Program Topic:** *Virtual Base Classes*

**Program Title:**

Write a C++ program using virtual base classes.

**Objectives:**

To learn about virtual base classes.

Syntax:

class derived : virtual public base

{

// body

}

**Source Code:**

#include <iostream>

using namespace std ;

class base

{

public :

int i;

};

class derived1 : virtual public base

{

public :

int j;

};

class derived2 : virtual public base

{

public :

int k;

};

class derived3 : public derived1 , public derived2

{

public :

int product () { return i \* j \* k; }

};

int main ()

{

derived3 ob;

ob.i = 1;

ob.j = 2;

ob.k = 3;

cout << " Product = " << ob. product () << endl;

return 0;

}

**Output:**

Product = 6

Process returned 0 (0x0) execution time : 0.030 s

Press any key to continue.