* **Example 1: Whenever the derived class’s default constructor is called, the base class’s default constructor is called automatically.**

#include <iostream>

using namespace std;

class Account {

public:

float salary = 60000;

Account() {

cout << "hello base";

} };

class Programmer: public Account {

public:

float bonus = 5000;

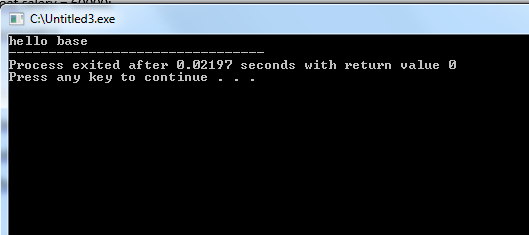
};

int main(void) {

Programmer p;

return 0;

}



**Example 2:**

#include <iostream>

using namespace std;

class Account {

public:

float salary = 60000;

Account(int i) {

cout << "hello";

} };

class Programmer: public Account {

public:

float bonus = 5000;

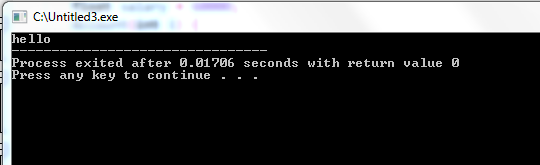
};

int main(void) {

Account a(5);

return 0;

}



* **Example 3: Code Example for single**

#include <iostream>

using namespace std;

class Account {

public:

float salary = 60000;

Account(int i) {

cout << "hello base";

} };

class Programmer: public Account {

public:

float bonus = 5000;

};

int main(void) {

Programmer p; // [Error] no matching function for call to 'Account::Account()'

return 0;

}

**Example 4:**

#include <iostream>

using namespace std;

class Account {

public:

float salary = 60000;

Account(int i) {

cout << "hello base";

} };

class Programmer: public Account {

public:

float bonus = 5000;

};

int main(void) {

Account a(2);

Programmer p; // [Error] no matching function for call to 'Account::Account()'

return 0;

}

**Example 5:**

#include <iostream>

using namespace std;

class Account {

public:

float salary = 60000;

Account(int i) {

cout << i;

} };

class Programmer: public Account {

public:

float bonus = 5000;

Programmer(int t):Account(t) {

}

};

int main(void) {

Programmer p1(5);

cout<< endl;

Account a(10);

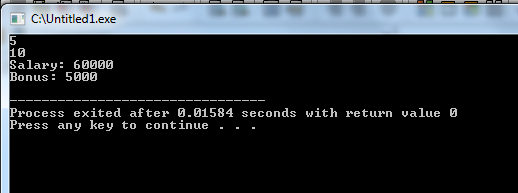
cout<< endl;

cout<<"Salary: "<<p1.salary<<endl;

cout<<"Bonus: "<<p1.bonus<<endl;

return 0;

}



**Another way using default values in base class constructor :**

#include <iostream>

using namespace std;

class Account {

public:

float salary = 60000;

Account(int I = 100) {

cout << "hello base";

} };

class Programmer: public Account {

public:

float bonus = 5000;

};

int main(void) {

Programmer p; // no error

return 0;

}

Graphical user interface, text

Description automatically generated

* **Code Example for multi level**

#include <iostream>

using namespace std;

class A{

public:

A(int a=1) { cout << "A()" << a << endl; }};

class B : public A{

public:

B(int b , int valA ):A(valA) { cout << "B()" << b << endl; } };

class C : public B{

public:

C(int c ,int valB, int ia):B(valB ,ia) { cout << "C()" <<c << endl; } };

int main(){

C c(2,4,6);

return 0;}

**Remember you can not call your indirect base class directly**

**e.g.**

#include <iostream>

using namespace std;

class A{

public:

A(int a=1) { cout << "A()" << a << endl; }};

class B : public A{

public:

B(int b , int valA ):A(valA) { cout << "B()" << b << endl; } };

class C : public B{

public:

C(int c ,int valB, int ia):B(valB ,ia**), A(ia)** { cout << "C()" <<c << endl; } };

int main(){

C c(2,4,6);

return 0;}

Graphical user interface, text, application, Teams

Description automatically generated

**Syntax for multiple inheritance**

**Derived class constructor(arg1,arg2,….argn) : base1(arg1) , base2(arg2){**

**D= argn;**

**}**

* **OR**
* **Derived class constructor(arg1,arg2,arg3) : base1(arg1) , base2(arg2), derived(arg3){}**

**Code Example**

#include <iostream>

using namespace std;

Class A{

Int a;

Public:

A(int x){

a=x;

Cout<< “A constructor”;

}

};

Class B{

Int b;

Public:

B(int y){

b=y;

Cout<< “B constructor”;

}

};

Class C : public A , public B{

Int m;

Public:

C(int x, int y, int z): A(x) , B (y){

m=z;

Cout<< “c constructor”;

}

};

#include <iostream>

using namespace std;

#include  "C.h"

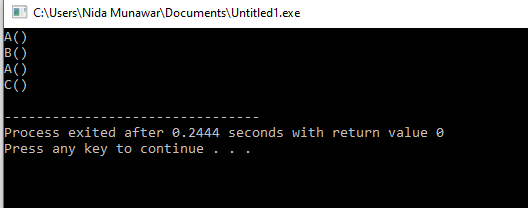
int main(){

C obj(10,20,30);

}

**Constructor and destructor in Hierarchical inheritance**

* **#include <iostream>**
* **using namespace std;**
* **class A{**
* **public:**
* **A() { cout << "A()" << endl; }};**
* **class B : public A{**
* **public:**
* **B() { cout << "B()" << endl; } };**
* **class C : public A{**
* **public:**
* **C() { cout << "C()" << endl; } };**
* **int main(){**
* **B b;**
* **C c;**
* **return 0;}**

****