**C++ Inheritance & Aggregation**

In C++, inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such way, you can reuse, extend or modify the attributes and behaviors which are defined in other class.

In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

Advantage of C++ Inheritance

**Code reusability:** Now you can reuse the members of your parent class. So, there is no need to define the member again. So less code is required in the class.

|  |  |
| --- | --- |
| **Single Level Inheritance Example: Inheriting Fields**  When one class inherits another class, it is known as single level inheritance. | C++ Single Level Inheritance Example: Inheriting Methods |
| #include<iostream>  using namespace std;  class animalTop{  public:  int salary = 500;  };  class dog : public animalTop{  public:  int bonus = 2;  };  int main(){  dog d;  cout << "Salary : " << d.salary << endl;  cout << "bonus : " << d.bonus << endl;  int total = d.salary + d.bonus;  cout << "total : " << total;  return (0);  }  /\*Salary : 500  bonus : 2  total : 502\*/ | #include <iostream>  using namespace std;  class Animal {  public:  void eat() {  cout<<"Eating..."<<endl;  }  };  class Dog: public Animal  {  public:  void bark(){  cout<<"Barking...";  }  };  int main(void) {  Dog d1;  d1.eat();  d1.bark();  return 0;  }  /\*Eating...  Barking...\*/ |

## C++ Multi Level Inheritance Example

## When one class inherits another class which is further inherited by another class, it is known as multi level inheritance in C++. Inheritance is transitive so the last derived class acquires all the members of all its base classes.

#include <iostream>

using namespace std;

class Animal {

public:

void eat() {

cout<<"Eating..."<<endl;

}

};

class Dog: public Animal{

public:

void bark(){

cout<<"Barking..." << endl;

}

};

class babyDog : public Dog{

public:

void weep(){

cout << "weeping...." << endl;

}

};

int main(void) {

babyDog bd;

bd.eat() ;

bd.bark();

bd.weep();

return 0;

}

/\*Eating...

Barking...

weeping....\*/

# C++ Aggregation (HAS-A Relationship)

In C++, aggregation is a process in which one class defines another class as any entity reference. It is another way to reuse the class. It is a form of association that represents HAS-A relationship.

## C++ Aggregation Example

Let's see an example of aggregation where Employee class has the reference of Address class as data member. In such way, it can reuse the members of Address class.

***Example:***

|  |
| --- |
| #include <iostream>  using namespace std;  class Address{  public:  string road, area, town;  public:  Address(string rd, string ar, string tw){  this->road = rd;  this->area = ar;  this->town = tw;  }  };  class Employee{  string name;  Address \*address;  int salary;    public:  Employee(string nm, int sal, Address \*address){  this->name = nm;  this->salary = sal;  this->address = address;  }    void display(){  cout << "name : " << name << " Salary : " << salary << " road : " << address->road << " area : " << address->area << " town : "  << address->town << endl;  }  };  int main(){  Address a = Address("fuck road", "pornhub", "BDSM");  Employee e = Employee("Snow", 5800, &a);  e.display();  return(0);  }  //name : Snow Salary : 5800 road : fuck road area : pornhub town : BDSM |