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 the HAS-A relationship.

What is Aggregation in C++?

In C++, Aggregation is used to represent the 'HAS-A' relationship between two objects. It is a type of association. If in a process, one class defines another class as any entity reference then it is known as Aggregation. With the help of aggregation, you can also reuse the class.

An object should define the following relationships to qualify as an aggregation:-

- Must be a part of the class.
- The member can belong to one or more classes at a time.
- Member is unknown about the existence of the object.
- The relationship is unidirectional.

NOTE:- For creating or destroying the members or parts, Aggregation is not responsible.

Syntax:

```
Class ClassPart
{
//instance variables
//instance methods
}
class Tech
{
ClassPart* partclass;
}
```

In the above, the Tech class represents the class that is a container class for other ClassPart. And it is included in the object of the Tech class. The Tech class's each object holds a reference pointer to the object of the ClassPart.

Advantages of Aggregations in C++

- Aggregation represents unidirectional relation between the objects of 2 classes. It is a one-direction relation.
- Aggregation also helps in improving readability and reusability of the program code.

- It helps in initiating the relation between objects of 2 classes where one class is the Whole class and other is the part of the class. It is useful in representing the HAS-A relation.
- Aggregation helps in making your program code more readable and understandable to represent the relation.

C++ Aggregation Example

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

```
#include <iostream>
using namespace std;
class Address {
    public:
   string addressLine, city, state;
     Address(string addressLine, string city, string state)
    {
        this->addressLine = addressLine;
        this->city = city;
        this->state = state;
    }
};
class Employee
    {
        private:
        Address* address;  //Employee HAS-A Address
        public:
        int id;
        string name;
        Employee(int id, string name, Address* address)
       {
           this->id = id;
           this->name = name;
           this->address = address;
       }
     void display()
       {
           cout<<id <<" "<<name<< " "<<
             address->addressLine<< " "<< address->city<< " "<<address->state<<endl;
       }
   };
int main(void) {
    Address a1= Address("C-146, Sec-15","Noida","UP");
```

```
34.     Employee e1 = Employee(101,"Nakul",&a1);
35.             e1.display();
36.    return 0;
37. }
```

Output:

101 Nakul C-146, Sec-15 Noida UP

Example:

```
#include <iostream.h>

using namespace std;

class Student

{

public:

string Name;

float ID;

Student(string Name, float ID)

{

this-> Name = Name;

this-> ID = ID;

}

};

class Details

{

private:

Student* std;

public:
```

```
int stdclass;
string section;
Details(int stdclass, string section, Student* std)
{
this->stdclass = stdclass;
this->section = section;
this->std = std;
}

void detail()
{
cout << "Student Class : " << stdclass << endl;
cout << "Student Section : " << section << endl;
cout << "Student Name : " << std->Name<< endl;
cout << "Student ID : "<< std->ID << endl;
}
};
int main()
{
Student s1 = Student("Khush", 1);
Details d1 = Details(6, "A", &s1);
d1.detail();
return 0;
}
```

Output

```
Student Class : 6

Student Section : A

Student Name : Khush

Student ID : 1
```