10.1 — Object relationships

BY ALEX ON AUGUST 10TH, 2016 | LAST MODIFIED BY ALEX ON JANUARY 23RD, 2020

Life is full of recurring patterns, relationships, and hierarchies between objects. By exploring and understanding these, we can gain insight into how real-life objects behave, enhancing our understanding of those objects.

For example, let’s say one day you’re walking down the street, and you see a bright yellow object attached to a green shrubby object. You’d probably recognize that the bright yellow thing is a flower, and the green shrubby thing is a plant. Even though you’d never seen this particular type of plant before, you’d know that the green things are leaves, collecting sunlight. You’d know that the flower helps the plant propagate itself. You’d also know that if you killed the plant, the flower would die too.

But how can you know all of this without ever encountering a plant of this type before? You know this because you understand the abstract concept of plants, and recognize that this plant is an instantiation of that abstraction. You know that most plants are composed (in part) of leaves, and some have flowers. You know that the leaves interact with the sunlight (even if you don’t know how, exactly), and that the flower’s existence depends on the plant. Because you know all of these things about plants in general, you can infer a lot about this plant.

Similarly, programming is also full of recurring patterns, relationships and hierarchies. Particularly when it comes to programming objects, the same patterns that govern real-life objects are applicable to the programming objects we create ourselves. By examining these in more detail, we can better understand how to improve code reusability and write classes that are more extensible.

In previous chapters, we’ve already explored some ideas around recurring patterns: we’ve created loops and functions to allow us to do a particular task many times. Additionally, we’ve created our own enums, structs, and classes to allow us to instantiate objects of a given type.

We’ve also explored some primitive forms of hierarchy, such as arrays (which allow us to group elements into a larger structure) and recursion, where a function calls a derivative version of itself.

However, we haven’t yet focused much on the relationship between objects, particularly as it relates to programming.

Relationships between objects

There are many different kinds of relationships two objects may have in real-life, and we use specific “relation type” words to describe these relationships. For example: a square “is-a” shape. A car “has-a” steering wheel. A computer programmer “uses-a” keyboard. A flower “depends-on” a bee for pollination. A student is a “member-of” a class. And your brain exists as “part-of” you (at least, we can reasonably assume so if you’ve gotten this far).

All of these relation types have useful analogies in C++.

In this chapter, we’ll explore the nuances of the relation types “part-of”, “has-a”, “uses-a”, “depends-on”, and “member-of”, and show how they can be useful in the context of C++ classes. We’ll also explore a couple of related topics that don’t fit nicely anywhere else.

Then we’ll devote the following two chapters to exploring “is-a” relationships, via C++’s inheritance model and virtual functions. Yup, it’s a biggie.

Alright, enough context setting. Let’s get to it.

10.2 -- Composition

Index

9.x -- Chapter 9 comprehensive quiz

C++ TUTORIAL | PRINT THIS POST

15 comments to 10.1 — Object relationships

hellmet
November 21, 2019 at 4:59 am · Reply

I am a C++ learner, a growing tree, increasing my knowledge under the sunlight of this website and the kind help from Alex and Nascardriver.! This site "is-a"mazing!

ConfusedKid
February 21, 2019 at 3:10 am · Reply

Okay, let's break it down a bit:
Car is a vehicle;
Car has a steering wheel;
Car uses a gas station;
Car depends on fuel;
Car is a member of legendary items;
Correct me if I'm wrong but what is the difference between "is a" and "a member of".

Alex
February 22, 2019 at 12:52 pm · Reply

X is-a Y denotes that X is a subtype of Y. A car is a vehicle.

X is a member of Y denotes that X is a part of some container. A car is a member of the shipping container is it inside of.

Ganesh Bhadane
January 11, 2019 at 2:50 am · Reply

Hi Alex,

What are the differences between object and instance?

Alex
January 12, 2019 at 10:57 pm · Reply

In C++ the terms are synonymous.

DonKillha
November 14, 2018 at 11:32 am · Reply

And your brain exists as "part-of" you (at least, we can reasonably assume so if you've gotten this far).

I see you with them jokes xD

Ryder
February 15, 2018 at 5:06 pm · Reply

How many types of relations between two objects exists?

Can we divide those relations into major groups?

These are your examples:"part-of", "has-a", "uses-a", "depends-on", and "member-of"...

Emm, what about the following relations?

1. study something
2. work for somebody
3. look for something
4. listen to something

...

Alex
February 15, 2018 at 6:08 pm · Reply

These things you list are actions, not relationships.

For example, "study something". Studying is an action. But who is doing the studying and what are they studying?

Once you know that, you can frame this as a relationship. You (object) uses-a (relationship) programming book (object). Now we have a relationship between objects.

Ryder
February 17, 2018 at 8:37 am · Reply

It seems that the types of "relations" are too many. In other word, it is depends on a specific "frame".

You can always find two objects for a verb.

1. study something: I study your website.
2. work for somebody: I work for your website.
3. look for something: I look for information.
4. listen to something: I listen to your suggestions.

The relation is between "subject" and "object" in the above examples.

I think the important thing is to study "inheritance model and virtual functions" within C++. To learn how C++ implements a type of "relation", and how this implementation reduce repeat works is the key things in this chapter.

James
August 1, 2017 at 9:59 am · Reply

These tutorials have been brilliant so far. I have learnt so much. Thank you!

Zik-sin
March 5, 2017 at 9:49 am · Reply

Thank you for your great tutorials so far. Better than any of my CS courses in U of *. Thanks !

Zachary Fojtasek
January 11, 2017 at 11:40 am · Reply

Is there a "shares a ____ with" section?

example: the kitchen shares a door with the hallway

Alex
January 16, 2017 at 6:11 pm · Reply

Not specifically, but this could be represented as an aggregation, where both Kitchen and Hallway have a Door (that neither own).

Andrew42
September 27, 2016 at 7:39 am · Reply

Wow, the first comment. Alex, please fix the typo in "Additionaally".

Thanks for what you do.

Alex
September 27, 2016 at 10:12 am · Reply

Fixed! Thanks for pointing that out.