

12.1 — Introduction to compound data types

👤 **ALEX¹** ⌚ **NOVEMBER 23, 2024**

In lesson [4.1 -- Introduction to fundamental data types](https://www.learncpp.com/cpp-tutorial/introduction-to-fundamental-data-types/) (<https://www.learncpp.com/cpp-tutorial/introduction-to-fundamental-data-types/>)², we introduced the fundamental data types, which are the basic data types that C++ provides as part of the core language.

We've made much use of these fundamental types in our programs so far, especially the `int` data type. And while these fundamental types are extremely useful for straightforward uses, they don't cover our full range of needs as we begin to do more complicated things.

For example, imagine you were writing a math program to multiply two fractions. How would you represent a fraction in your program? You might use a pair of integers (one for the numerator, one for the denominator), like this:

```
1  #include <iostream>
2
3  int main()
4  {
5      // Our first fraction
6      int num1 {};
7      int den1 {};
8
9      // Our second fraction
10     int num2 {};
11     int den2 {};
12
13     // Used to eat (remove) the slash between the numerator and denominator
14     char ignore {};
15
16     std::cout << "Enter a fraction: ";
17     std::cin >> num1 >> ignore >> den1;
18
19     std::cout << "Enter a fraction: ";
20     std::cin >> num2 >> ignore >> den2;
21
22     std::cout << "The two fractions multiplied: "
23         << num1 * num2 << '/' << den1 * den2 << '\n';
24
25     return 0;
26 }
```

And a run of this program:

```
Enter a fraction: 1/2
Enter a fraction: 3/4
The two fractions multiplied: 3/8
```

While this program works, it introduces a couple of challenges for us to improve upon. First, each pair of integers is only loosely linked -- outside of comments and the context of how they are used in the code, there's little to suggest that each numerator and denominator pair are related. Second, following the DRY

(don't repeat yourself) principle, we should create a function to handle the user inputting a fraction (along with some error handling). However, functions can only return a single value, so how would we return the numerator and denominator back to the caller?

Now imagine another case where you're writing a program that needs to keep a list of employee IDs. How might you do so? You might try something like this:

```
1 | int main()
2 | {
3 |     int id1 { 42 };
4 |     int id2 { 57 };
5 |     int id3 { 162 };
6 |     // and so on
7 | }
```

But what if you had 100 employees? First, you'd need to type in 100 variable names. And what if you needed to print them all? Or pass them to a function? We'd be in for a lot of typing. This simply doesn't scale.

Clearly fundamental types will only carry us so far.

Compound data types

Fortunately, C++ supports a second set of data types: **compound data types** (also sometimes called **composite data types**) are types that are defined in terms of other existing data types. Compound data types have additional properties and behaviors that make them useful for solving certain types of problems.

Key insight

Every data type is either a fundamental type or a compound type. The C++ language standard explicitly defines which category each type falls into.

As we'll show in this chapter and future chapters, we can use compound data types to elegantly solve all of the challenges we presented above.

C++ supports the following compound types:

- Functions
- C-style Arrays
- Pointer types:
 - Pointer to object
 - Pointer to function
- Pointer to member types:
 - Pointer to data member
 - Pointer to member function
- Reference types:
 - L-value references
 - R-value references
- Enumerated types:
 - Unscoped enumerations
 - Scoped enumerations

- **Class types:**
 - **Structs**
 - **Classes**
 - **Unions**

You've already been using one compound type regularly: functions. For example, consider this function:

```
1 void doSomething(int x, double y)
2 {
3 }
```


The type of this function is `void(int, double)`. Note that this type is composed of fundamental types, making it a compound type. Of course, functions also have their own special behaviors as well (e.g. being callable).

Because there's a lot of material to cover here, we'll do it over multiple chapters. In this chapter, we'll cover some of the more straightforward compound types, including `1-value references`, and `pointers`. Next chapter, we'll cover `unscoped enumerations`, `scoped enumerations`, and our first class type: `structs`. Then, in the chapters beyond that, we'll introduce classes and dig into some of the more useful `array` types. This includes `std::string` (introduced in lesson [5.7 -- Introduction to std::string](#) (<https://www.learncpp.com/cpp-tutorial/introduction-to-stdstring/>³), which is actually a class type!

Nomenclature


A **class type** is a type that is a struct, class, or union. We'll use this term a lot in future lessons.

Got your game face on? Let's go!



[Next lesson](#)
12.2 [Value categories \(lvalues and rvalues\)](#)

4



[Back to table of contents](#)

5



[Previous lesson](#)
F.X [Chapter F summary and quiz](#)

6

7

Leave a comment...

 Name*

 Email*




Notify me about replies:



POST COMMENT

 Find a mistake? Leave a comment above!?

 Avatars from <https://gravatar.com/>⁹ are connected to your provided email address.

44 COMMENTS

Newest ▼



RURU

 June 18, 2025 9:36 pm PDT

best of luck yll for sticking this far and from now on the chapter are gonna be fire !!!!!!!!!!!
The real deal starts now

 3  Reply



Jacob

 May 21, 2025 3:26 pm PDT

Typo: "... we'll introduce classes ..."

 0  Reply

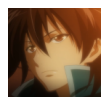


RSH

 May 11, 2025 2:43 am PDT

Finally been wanting to learn this in specfici

 0  Reply



Zeca

 April 21, 2025 12:09 pm PDT

lets gooooo

 0  Reply



Choco

 April 17, 2025 2:54 pm PDT

lets go, two month i will be rich off my homemade malware that i send to old people :D

 5  Reply

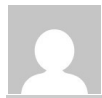


jimothy

 Reply to [Choco](#)¹⁰  May 11, 2025 9:53 am PDT

LMAO

 0  Reply



Max

 April 13, 2025 2:59 pm PDT

We out here

 0  Reply



BiscuitRE-L

 February 20, 2025 11:31 pm PST

Lock in gang

 20  Reply



David Pinheiro

 November 8, 2024 9:45 am PST

Arrays here means "C-Style Arrays" right?

 0  Reply



Alex

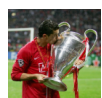
Author

 Reply to [David Pinheiro](#)¹¹  November 10, 2024 12:01 pm PST


Yes. Changed the text to reflect this.

std::array and std::vector are class types (implemented using C-style arrays).

 2  Reply



abdullah hatem

 July 8, 2024 2:11 pm PDT

this chapter gonna be fire !

 18  Reply



PooperShooter

Reply to [abdullah hatem](#)¹² ⌚ December 28, 2024 5:35 pm PST

honestly the next couple are gonna be real good all the way up to and including vectors

👍 1

➡ Reply



Priya

⌚ October 15, 2023 1:32 am PDT

Images are taking so much time to load some are not even loading for about 1 hour.

Contents are great.

Please try to deliver in more presentable way (Like gfg, W3).

👍 1

➡ Reply



NordicPeace

Reply to [Priya](#)¹³ ⌚ November 30, 2024 4:41 am PST

?? where are pictures on this web ?

👍 5

➡ Reply



GalacticDaddy

Reply to [NordicPeace](#)¹⁴ ⌚ February 25, 2025 11:18 am PST

he proolly meant ADS

👍 0

➡ Reply



NordicCat

Reply to [GalacticDaddy](#)¹⁵ ⌚ March 2, 2025 11:43 pm PST

I have an ad blocker so no ads for me.

👍 3

➡ Reply



just_a_guy

Reply to [NordicCat](#)¹⁶ ⌚ May 15, 2025 2:55 am PDT

it would be better if you turn it off because this is the only way to support him

👍 0

➡ Reply

Links

1. <https://www.learncpp.com/author/Alex/>
2. <https://www.learncpp.com/cpp-tutorial/introduction-to-fundamental-data-types/>
3. <https://www.learncpp.com/cpp-tutorial/introduction-to-stdstring/>
4. <https://www.learncpp.com/cpp-tutorial/value-categories-lvalues-and-rvalues/>
5. <https://www.learncpp.com/>
6. <https://www.learncpp.com/cpp-tutorial/chapter-f-summary-and-quiz/>
7. <https://www.learncpp.com/introduction-to-compound-data-types/>
8. https://www.learncpp.com/cpp-tutorial/introduction-to-type-conversion-and-static_cast/
9. <https://gravatar.com/>
10. <https://www.learncpp.com/cpp-tutorial/introduction-to-compound-data-types/#comment-609340>
11. <https://www.learncpp.com/cpp-tutorial/introduction-to-compound-data-types/#comment-603957>
12. <https://www.learncpp.com/cpp-tutorial/introduction-to-compound-data-types/#comment-599338>
13. <https://www.learncpp.com/cpp-tutorial/introduction-to-compound-data-types/#comment-588837>
14. <https://www.learncpp.com/cpp-tutorial/introduction-to-compound-data-types/#comment-604707>
15. <https://www.learncpp.com/cpp-tutorial/introduction-to-compound-data-types/#comment-608061>
16. <https://www.learncpp.com/cpp-tutorial/introduction-to-compound-data-types/#comment-608207>
17. <https://g.ezoic.net/privacy/learncpp.com>