1.3 — Introduction to variables

BY ALEX ON MAY 30TH, 2007 | LAST MODIFIED BY ALEX ON JULY 8TH, 2019

Data

In lesson **1.1 -- Statements and the structure of a program**, you learned that the majority of instructions in a program are statements, and that statements are grouped into functions. These statements perform actions that (hopefully) generate whatever result the program was designed to produce.

But how do programs actually produce results? They do so by manipulating (reading, changing, and writing) data. In computing, **data** is any information that can be moved, processed, or stored by a computer.

Key insight

Programs are collections of instructions that manipulate data to produce a desired result.

A program can acquire data to work with in many ways: from a file or database, over a network, from the user providing input on a keyboard, or from the programmer putting data directly into the source code of the program itself. In the "Hello world" program from the aforementioned lesson, the text "Hello world!" was inserted directly into the source code of the program, providing data for the program to use. The program then manipulates this data by sending it to the monitor to be displayed.

Data on a computer is typically stored in a format that is efficient for storage or processing (and is thus not human readable). Thus, when the "Hello World" program is compiled, the text "Hello world!" is converted into a more efficient format for the program to use (binary, which we'll discuss in a future lesson).

Objects and variables

All computers have memory, called **RAM** (short for random access memory), that is available for your programs to use. You can think of RAM as a series of mailboxes that can be used to hold data while the program is running. A single piece of data, stored in memory somewhere, is called a **value**.

In some older programming languages (like Apple Basic), you could directly access these mailboxes (a statement could say something like *go get the value stored in mailbox number 7532*).

In C++, direct memory access is not allowed. Instead, we access memory indirectly through an object. An **object** is a region of storage (usually memory) that has a value and other associated properties (that we'll cover in future lessons). When an object is defined, the compiler automatically determines where the object will be placed in memory. As a result, rather than say *go get the value stored in mailbox number 7532*, we can say, *go get the value stored by this object* and the compiler knows where in memory to look for that value. This means we can focus on using objects to store and retrieve values, and not have to worry about where in memory they're actually being placed.

Objects can be named or unnamed (anonymous). A named object is called a **variable**, and the name of the object is called an **identifier**. In our programs, most of the objects we create will be variables.

Author's note

In general programming, the term *object* typically refers to a variable, data structure in memory, or function. In C++, the term *object* has a narrower definition that excludes functions.

Variable instantiation

In order to create a variable, we use a special kind of declaration statement called a **definition** (we'll clarify the difference between a declaration and definition later).

Here's an example of defining a variable named x:

```
1 int x; // define a variable named x, of type int
```

At compile time, when the compiler sees this statement, it makes a note to itself that we are defining a variable, giving it the name *x*, and that it is of type *int* (more on types in a moment). From that point forward (with some limitations that we'll talk about in a future lesson), whenever the compiler sees the identifier *x*, it will know that we're referencing this variable.

When the program is run (called **runtime**), the variable will be instantiated. **Instantiation** is a fancy word that means the object will be created and assigned a memory address. Variables must be instantiated before they can be used to store values. For the sake of example, let's say that variable *x* is instantiated at memory location 140. Whenever the program then uses variable x, it will access the value in memory location 140. An instantiated object is sometimes also called an **instance**.

Data types

So far, we've covered that variables are a named region of storage that can store a data value (how exactly data is stored is a topic for a future lesson). A **data type** (more commonly just called a **type**) tells the compiler what type of value (e.g. a number, a letter, text, etc...) the variable will store.

In the above example, our variable *x* was given type *int*, which means variable x will represent an integer value. An **integer** is a number that can be written without a fractional component, such as 4, 27, 0, -2, or -12. For short, we can say that *x* is an *integer variable*.

In C++, the type of a variable must be known at **compile-time** (when the program is compiled), and that type can not be changed without recompiling the program. This means an integer variable can only hold integer values. If you want to store some other kind of value, you'll need to use a different variable.

Integers are just one of many types that C++ supports out of the box. For illustrative purposes, here's another example of defining a variable using data type *double*:

```
1 double width; // define a variable named width, of type double
```

C++ also allows you to create your own user-defined types. This is something we'll do a lot of in future lessons, and it's part of what makes C++ powerful.

For these introductory chapters, we'll stick with integer variables because they are conceptually simple, but we'll explore many of the other types C++ has to offer soon.

Defining multiple variables

It is possible to define multiple variables *of the same type* in a single statement by separating the names with a comma. The following 2 snippets of code are effectively the same:

```
1 int a;
2 int b;
```

is the same as:

```
int a, b;
```

When defining multiple variables this way, there are two common mistakes that new programmers tend to make (neither serious, since the compiler will catch these and ask you to fix them):

The first mistake is giving each variable a type when defining variables in sequence.

```
int a, int b; // wrong (compiler error)

int a, b; // correct
```

The second mistake is to try to define variables of different types in the same statement, which is not allowed. Variables of different types must be defined in separate statements.

```
int a, double b; // wrong (compiler error)

int a; double b; // correct (but not recommended)

// correct and recommended (easier to read)
int a;
double b;
```

Best practice

Although the language allows you to do so, avoid defining multiple variables in a single statement (even if they are the same type). Instead, define each variable in a separate statement (and then use a single-line comment to document what it is used for).

Summary

In C++, we use variables to access memory. Variables have an identifier, a type, and a value (and some other attributes that aren't relevant here). A variable's type is used to determine how the value in memory should be interpreted.

In the next lesson, we'll look at how to give values to our variables and how to actually use them.

Quiz time

Question #1

What is data?

Show Solution

Question #2

What is a value?

Show Solution

Question #3

What is a variable?

Show Solution

Question #4

What is an identifier?

Show Solution

Question #5

What is a type?

Show Solution

Question #6

What is an integer?

Show Solution

1.4 -- Variable assignment and initialization

Index

1.2 -- Comments

C++ TUTORIAL | PRINT THIS POST

380 comments to 1.3 — Introduction to variables

« Older Comments 1 ... 4 5 6

Amandeep Singh
January 31, 2020 at 5:27 am · Reply

brother everything fine but u should explain about unnamed and named object with example!

prince
January 9, 2020 at 4:35 am · Reply

I am a little bit confuse on what is an identifier and a variable.

examaple:

int math = 0

can someone please tell me where is the identifier and the variable based on my example.

thanks.

nascardriver
January 9, 2020 at 4:39 am · Reply

`math` is a variable, that includes the type, identifier, value, and more. The identifier is only the name of the variable, "math".

Harshitha
December 27, 2019 at 6:27 am · Reply

Sir , what does it mean to be "defining a variable" when the meaning of variable is "a named object" The statements contrasting in the above paragraph of variable instantiation.

nascardriver
December 28, 2019 at 4:45 am · Reply

The definition of a variable is what creates its memory and optionally sets an initial value. Variables can also be declared, which tells the compiler that the variable exists, but it doesn't create the variable. This is covered later.

Christian
October 11, 2019 at 3:44 am · Reply

What kind of program is you trying to do in this lesson?

papa smurf
October 16, 2019 at 7:31 pm · Reply

It's an example program to show errors made when creating variables.

Yiğit

August 25, 2019 at 5:50 am · Reply

I have a question, your quizzes are kinda hard I mean I take notes too but I don't know In thıs chapter, quizzes chapter I just did correct what's an ınteger, and your explanation is so techical I am new at coding I know variables just containers they can hold values but you are saying "A variable is a named region of memory." And why we need to learn a lot of details??? Just asking :=) And can you give me c++ challenge website or something I can't found (I mean hackerrank ok but hard, you think I should do hard or easy?)

nascardriver
August 25, 2019 at 6:11 am · Reply

You need to know the details if you want to be good at the language.
There's no point in trying challenges yet, you're just learning the basics. When you're done with these tutorials, you can start at easy challenges. If you find them too easy, move up.

Yiğit
August 25, 2019 at 7:13 am · Reply

idk maybe my fault i can't focus or something but 1.5 take input from user part i answered all of the quiz questions and I did all of them corectly maybe this is just my fault i didn't understand this is idk but amazing tutorail thx ALEX AND OTHERS WHO İS HELPİNG US <3

Thomas Kort
August 28, 2019 at 12:20 pm · Reply

@Yiğit Hey! this is a good website to do easy challenges. It's kind of the same as this, but it's less detaild and with less of the "best practice". if link does'nt work you can google w3schools and go to c++.
https://www.w3schools.com/cpp/default.asp

Yiğit
August 30, 2019 at 3:59 pm · Reply

Thank you I will check

Elyes
August 5, 2019 at 10:56 am · Reply

Hello, I'm very knew to coding and programming. I've started since the beginning of this tutorial that was recommended by a friend of mine. My question is : If I didn't understand some things in some lessons, do I have to worry or will learn through the tutorials?
Second question : IF I ever (hopefully) finished the tutorials and learned this programming langauge that is the first one I'm learning, what does it serve to do, like developing websites, coding in games if not these than what?
Thank you very much.

nascardriver
August 6, 2019 at 2:53 am · Reply

There are quizzes at the end of some lessons, you should be able to do those. If you're stuck, re-visit the relevant lessons and try to understand them. If you don't, ask.
C++ isn't a language for web-development, you'll need javascript/typescript, html and css for that.
C++ is meant to be used for native applications, ie. everything that runs on your computer, including video games. It can also be used for hardware development, eg. through Arduino.

Elyes
August 7, 2019 at 10:10 am · Reply

A friend told me that If I learn this language It will be so much easier to learn other languages since most of them are C based including javascript.

nascardriver
August 7, 2019 at 10:52 pm · Reply

Your friend is absolutely right. C++ covers a lot of features and concepts that are present in other languages as well. Knowing C++ will help you big time if you want to learn another language.

Elyes
August 8, 2019 at 1:36 am · Reply

You're very helpful, thanks very much.

VichTrogdor
July 6, 2019 at 10:23 am · Reply

In question 2:
"A value is a single piece of data stored in memory."
should be
"A value is a single piece of datum stored in memory."

Alex
July 8, 2019 at 10:50 am · Reply

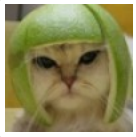
Datum is an archaic word that is generally less preferred than use of "data" as a singular in modern vernacular. Even the statistics website FiveThirtyEight doesn't use the word "datum". So although you're not wrong, we'll favor the modern usage here.

Vitaliy Sh.
January 4, 2020 at 2:31 am · Reply

https://en.wikipedia.org/wiki/FiveThirtyEight.
Funny enough: The Disney has them all.

VichTrogdor
July 6, 2019 at 10:09 am · Reply

"An object is region of storage" should be "An object is a region of storage"
Just an error in the text that could be fixed.



Alex
July 8, 2019 at 10:37 am · Reply

Fixed. Thanks!

Daen
May 29, 2019 at 10:55 pm · Reply

I am confused by what you mean by -2 is short for -12 can some please explain why this is?

nascardriver
May 29, 2019 at 11:54 pm · Reply

"4, 27, 0, -2, or -12" are examples of integers. After that comes a new sentence.

Mabel
May 28, 2019 at 5:35 pm · Reply

Are variables basically a place where we store data - and the compiler keeps it somewhere in the computer memory?

nascardriver
May 29, 2019 at 2:06 am · Reply

yes

Amanda
May 27, 2019 at 8:59 am · Reply

Thank you!

Amanda
May 27, 2019 at 7:08 am · Reply

Are objects and variables technically the same thing?

nascardriver
May 27, 2019 at 8:17 am · Reply

yes

Amanda
May 27, 2019 at 8:59 am · Reply

Thank you!

Giselle Freude
July 16, 2019 at 5:34 pm · Reply

Objects and variables are NOT the same thing. An object can be a variable, but the reverse is not necessarily true. Variables are not always objects. An object is a term that means a group of data along with operations on that data all grouped together into one thing. I will give you a simple example. Your wallet could be an object. The "data" that makes up your wallet are the various coins and dollar bills inside your wallet. The operations on the data could include counting the money, or spending the money. I hope that helps.

nascardriver
July 17, 2019 at 4:48 am · Reply

They're not the same, but your explanation isn't correct.
An object can be created without creating a new variable and a variable can be created without creating a new object.

> Variables are not always objects
"A variable is introduced by the declaration of a reference other than a non-static data member or of an object."
n4820 § 6 (6)

> An object is a term that means a group of data along with operations on that data all grouped together into one thing.
"An object is created by a definition [...], by a new-expression [...], when implicitly changing the active member of a union [...], or when a temporary object is created [...]."
n4820 § 6.6.2 (1)

They're different things, but often times can be interchanged.

« Older Comments 1 ... 4 5 6