8.1 — Welcome to object-oriented programming

BY ALEX ON AUGUST 23RD, 2007 | LAST MODIFIED BY ALEX ON JANUARY 23RD, 2020

Back in lesson **1.3 -- Introduction to variables**, we defined an object in C++ as, "a piece of memory that can be used to store values". An object with a name is called a variable.

In traditional programming (what we've been doing prior to this point), programs are basically lists of instructions to the computer that define data (via objects) and then work with that data (via statements and functions). Data and the functions that work on that data are separate entities that are combined together to produce the desired result. Because of this separation, traditional programming often does not provide a very intuitive representation of reality. It's up to the programmer to manage and connect the properties (variables) to the behaviors (functions) in an appropriate manner. This leads to code that looks like this:

```
driveTo(you, work);
```

So what is object-oriented programming? As with many things, it is perhaps understood most easily through use of an analogy. Take a look around you -- everywhere you look are objects: books and buildings and food and even you. Objects have two major components to them: 1) A list of relevant properties (e.g. weight, color, size, solidity, shape, etc…), and 2) Some number of behaviors that they can exhibit (e.g. being opened, making something else hot, etc…). These properties and behaviors are inseparable.

Object-oriented programming (OOP) provides us with the ability to create objects that tie together both properties and behaviors into a self-contained, reusable package. This leads to code that looks more like this:

```
you.driveTo(work);
```

This not only reads more clearly, it also makes it clearer who the subject is (you) and what behavior is being invoked (driving somewhere). Rather than being focused on writing functions, we're focused on defining objects that have a well-defined set of behaviors. This is why the paradigm is called "object-oriented".

This allows programs to be written in a more modular fashion, which makes them easier to write and understand, and also provides a higher degree of code-reusability. These objects also provide a more intuitive way to work with our data by allowing us to define how we interact with the objects, and how they interact with other objects.

Note that OOP doesn't replace traditional programming methods. Rather, it gives you additional tools in your programming tool belt to manage complexity when needed.

Object-oriented programming also brings several other useful concepts to the table: inheritance, encapsulation, abstraction, and polymorphism (language designers have a philosophy: never use a small word where a big one will do). We will be covering all of these concepts in the upcoming tutorials over the next few chapters. It's a lot of new material, but once you've been properly familiarized with OOP and it clicks, you may never want to go back to pure traditional programming again.

Note that the term "object" is overloaded a bit, and this causes some amount of confusion. In traditional programming, an object is a piece of memory to store values. And that's it. In object-oriented programming, an "object" implies that it is both an object in the traditional programming sense, and that it combines both properties and behaviors. From this point forward, when we use the term "object", we'll be referring to "objects" in the object-oriented sense.

8.2 -- Classes and class members

Index

7.x -- Chapter 7 comprehensive quiz

C++ TUTORIAL | PRINT THIS POST

61 comments to 8.1 — Welcome to object-oriented programming

Ged
December 13, 2019 at 9:37 am · Reply

Just wanted to thank you for this awesome tutorial. And ask a few questions? Is there anything else that you would suggest learning before starting OOP that you didn't mention in the previous chapters?

Btw Found a funny story and thought I could share with you about a game dev that didn't know what for/while loop was, neither array, structs and a lot of other things. But he managed to create the game (the length of the code is crazy).
Here is the code - https://www.reddit.com/r/programminghorror/comments/4dguj8/dev_didnt_know_about_for_or_while_loops/
Here is the game that he created - http://store.steampowered.com/app/351150

So basically this says that if you know what an "int", "bool" and "if" is, you can do anything :D

nascardriver
December 13, 2019 at 9:50 am · Reply

The lessons are written so that you can read them from begin the end. Everything you need has been, or will be, covered.

Lord Voldemort
December 8, 2019 at 1:01 am · Reply

what does it mean- "never use a small word where a big one will do".

nascardriver
December 8, 2019 at 7:11 am · Reply

See this comment
https://www.learncpp.com/cpp-tutorial/81-welcome-to-object-oriented-programming/comment-page-1/#comment-256129

Andre
August 6, 2019 at 11:21 am · Reply

Hello, I want to thank you for what you guys have done, this website is really something wonderful. I wanted to ask you if you could tell me what websites or books could you recommend me for learning data structures in cpp (linked lists, queue, binary trees, etc.)?. I really want to take this next step but don't know the best resources to do so. I apologize for posting this comment in this section, but I didn't know how else to ask (sent e-mail already but didn't get a response).

Thank you very much.

Jay
October 24, 2019 at 9:36 am · Reply

Data structures : Varsha Patil
Its the one we use in college.
Has all you have mentioned (Queues, LLs and Stacks etc)

Darshan
May 13, 2019 at 6:37 am · Reply

Generally speaking, is it correct to say that everything inside a function is either a keyword or an object (of course excluding parenthesis)? Is an entity which is not a keyword an object (atleast upto this point)?

Alex
May 16, 2019 at 5:01 pm · Reply

No, operators are neither a keyword nor an object. User-defined types aren't keywords or objects either.

Erik
April 4, 2019 at 11:27 pm · Reply

Hello Alex, thanks for a great tutorial.

Regarding C++ and OOP, would you say it is viable when deploying on embedded systems, where performance constraints are a much more real concern than for PC development?

Alex

April 5, 2019 at 5:14 pm · Reply

Viable, definitely. The best choice? Not sure. I haven't worked in that industry and can't speak the particulars of that industry.

NXPY
March 29, 2019 at 9:47 am · Reply

Is OOP only beneficial for the programmer ? I believe it does not enhance memory or processing considering we are only organising functions and data into objects.

Alex
March 29, 2019 at 4:43 pm · Reply

You might think so at first, but consider some of the benefits of OOP: higher productivity, better maintainability, and higher quality code. While that's good for developers, that's just as good for customers. It means they get programs sooner, cheaper, and at higher quality than they might otherwise.

OOP programs may user more memory or run more slowly, but often the difference isn't substantive. And in cases where it is, some of the development time that was saved by using OOP can be invested in optimizing for performance.

NXPY
March 29, 2019 at 7:28 pm · Reply

Thank you for the reply Alex ! OOP seems pretty good now !

Lan
September 10, 2018 at 3:14 am · Reply

Hi
I encountered a problem at the 28th line

```
1   #include<iostream>
2   #include<cmath>
3   #include<iomanip>
4   //Calculate the distance 2 points A and B
5   class Point
6   {
7   private:
8       float x,y;
9   public:
10      void enterPoint();
11      int calculateDistance(Point B);
12  };
13  void Point::enterPoint()
14  {
15      std::cout << "Enter the coordinate point ( x, y):";
16      std::cin >> x >> y;
17  }
18  int Point::calculateDistance(Point B)
19  {
20      Point A;
21      return sqrt(pow((A.x - B.x), 2) + pow((A.y - B.y), 2));
22  }
```

```
int main()
{
    Point A, B;
    A.enterPoint();
    B.enterPoint();
    std::cout << "The distance 2 points A and B: " << std::setprecision(2)<< calculateDista
    return 0;
}
```

Thank you !!

Rommel Villon
September 7, 2018 at 3:41 am · Reply

Great learning site.
Wish you all the best in life and may you have tons of patience updating, answering comments, guiding and keeping this place you created.

ray
June 3, 2018 at 10:14 am · Reply

whats funny is. im completely new to programming (less than a week) . nevermind c++ (main interest are game design/hack) anyway. i got up to using classes/objects/constructors with thenewboston series before coming here, so they entire time i was reading and doing quizzes i was thinking how much easier most of this would be with creating separate classes for these programs lol.

Peter Baum
May 13, 2018 at 4:00 pm · Reply

Nice explanation for this chapter but there was one tiny phrase I didn't like: "once you've been properly familiarized with OOP, you'll likely never want to go back to pure traditional programming again." I think that is misleading. OOP is perfect for some things and completely inappropriate for others. I don't think you have to over sell OOP at this point. The advantages will be obvious soon enough. :)

Val
January 19, 2018 at 3:20 pm · Reply

Hello Alex, I wrote a few thoughts regarding the difference between traditional (procedural) and object-oriented programming.
Could you please look and tell me if I understand this difference correctly? Sorry if it's too naively.

```
// Two types of programming: traditional (procedural) and
// object-oriented.

// Let's say we have three objects:
// - Pencil,
// - Car,
// - Quadratic Equation.

// The first two are objects of the Real World,
// and the third is an abstract mathematical entity.

// Each of these three objects possesses:
// - data,
// - behaviours.

// All three objects are used in some abstract programm.
```

```

//////////////////////////////
// TRADITIONAL PROGRAMMING //
//////////////////////////////

// The traditional programming approach is to completely
// separate data and behaviours.

// Place data of all three objects into
// "data container - programm's storage",
// place behaviours of all three objects into
// "behaviours container - programm's functions
// (aka methods, subroutines)".

// STORAGE                   | FUNCTIONS
// -------------------------|-------------------------------
// Pencil::data              | Pencil::behaviours
// Car::data                 | Car::behaviours
// Quadratic Equation::data | Quadratic Equation::behaviours

// The global functions use data from global storage.
// Is it similar to the Real World? NO!

//////////////////////////////////
// OBJECT-ORIENTED PROGRAMMING //
//////////////////////////////////

// The object-oriented programming approach is to
// combine related data and behaviours and encapsulate
// them into special units called classes.

// CLASS Pencil       | CLASS Car       | CLASS Quadratic Equation
// -------------------|-----------------|-------------------------------
//                    |                 |
// STORAGE            | STORAGE         | STORAGE
//                    |                 |
// Pencil::data       | Car::data       | Quadratic Equation::data
//                    |                 |
// FUNCTIONS          | FUNCTIONS       | FUNCTIONS
//                    |                 |
// Pencil::behaviours | Car::behaviours | Quadratic Equation::behaviours

// The class functions use data from class storage.
// Is it similar to the Real World? At least, it's more realistic.

// The concrete instantiations of classes are objects.

// The class is a scheme.
// The object is a realization of this scheme.

// The objects can communicate with each other, for example,
// by using the functions or methods exposed to the public.

// For instance, the object of class Car uses the public methods
// of the object of class Quadratic Equation to perform some
// internal calculations.

// Implementation of internal structures of classes is subject to
// the traditional or procedural programming paradigm.
```

Alex

January 23, 2018 at 8:12 pm · Reply

Sounds generally correct to me!

Object oriented programming is a bit of a mental shift from traditional function-oriented programming, but once you get the hang of it it's hard to go back!

nik
October 18, 2017 at 3:23 am · Reply

Hello! Great tutorial! Do you plan on adding lessons about OOP design in general(how to design good classes and relationship among them)? That would be nice I guess.)Or if you don't contemplate that, can you suggest some good books,articles on this topic?

Alex
October 21, 2017 at 10:13 am · Reply

Check out chapter 10. I think it contains some of what you're talking about.

Shamlei
August 9, 2017 at 2:00 am · Reply

Hey Nice lessons !

Still feel like I'm kinda weak when it comes to understanding pointers and references, I know the stuff but it's not natural I would sometime use a value instead of a reference for example for a swap function and realize after it wouldn't work.

How can I make it stick in my head clearly ? I can finish most of the exercices on my own but when coding I'm not 100% sure of what I'm doing and sometimes there's trial and error involved, especially when references are involved.

Great website, keep up the great work !

Alex
August 9, 2017 at 4:52 pm · Reply

It's hard to answer this question since pointers and references are used in multiple context. Is your question focused around when to use pointers and references in the context of passing values to functions?

Shamlei
August 10, 2017 at 3:25 am · Reply

I don't exactly understand what happens when a variable is passed to a function that takes in references, like in the swap function.

What happens to the variable ? Does the function creates references automatically ?

I know when to use what, it's more like I don't know what exactly happens or why I would use it.

Alex
August 12, 2017 at 12:57 pm · Reply

Nothing happens to the variable argument. It stays as it is. When the function is called, the reference parameter is created, and set to reference the argument. Then, within the function, the reference parameter can be used as if it were the original argument. At the end of the function, the reference is destroyed, but the original argument remains. Note that because the reference is treated as if it were the original argument, any changes to the referenced value change the argument!

So basically, you should pass by reference when you:
* Want to have the function be able to modify the value of the argument
* Are passing a non-fundamental type, to avoid making a needless copy (for fundamental types, this doesn't matter since making a copy is cheap)

Shamlei
August 12, 2017 at 4:15 pm · Reply

Thanks.

Sihoo
May 13, 2017 at 10:00 am · Reply

Whoooah. Finally got past chapter seven and came to OOP. YAY!

gigawert
April 17, 2017 at 2:35 pm · Reply

"yoBack in lesson..." LOL

Alex
April 18, 2017 at 1:40 pm · Reply

Issue fixed, dawg.

garryg
April 15, 2017 at 7:21 am · Reply

Excellent! Clears up the muddied waters nicely. It seems that one needs to think in terms of Object-Oriented rather than just as an Object to be oriented. Think I get it.

Garry

garryg
April 13, 2017 at 1:47 pm · Reply

Hi Alex- In this brief introduction to object-oriented-programming you talk about objects a lot but as far as I can tell you never really define what an object is. You may have done this in a previous chapter and, if so I apologize for my ignorance. FWIW ,In his 'elementary' textbook [Programming Principles and Practice Using C++] Bjarne Stroustrup defines an object as follows: "An object is a region of memory with a type that specifies what kind of information can be placed in it." This seems pretty straight-forward but the next sentence baffles me a bit. "A named object is called a variable." Does that mean that functions, which I believe(?) also occupy a place in memory, are not objects. If they are also objects what is(are) their datatype(s)? Second; should we then consider functions to be variables?
Alex, you and Stroustrup write really cool (sometimes inscrutable to me) text. It's really helpful

to be able to cross-reference as well have this blog. Thanks
Garry

Alex
April 14, 2017 at 2:46 pm · Reply

The term "object" is used in two different contexts. In the non-OOP sense, an object is as Stroustrup says. However, when we use the term object in an OOP sense, we mean something more: an object that combines both properties and behaviors. I rewrote this lesson a bit to try and make it clearer how OOP differs from traditional programming, and also how the term "object" is overloaded a bit.

Mike R.
April 5, 2017 at 10:57 am · Reply

re: "...language designers have a philosophy: never use a small word where a big one will do)..."

Instead, maybe:

"...language designers have a philosophy: never use a big word where a diminutive one will do)..."

:o)

BTW: Great tutorial! Thanks!

ejaz ashraf
June 1, 2016 at 9:06 pm · Reply

Honorable sir i want to become a good programmer but my basic concepts are to week in c plus plus .kindly guide me

Izzy
July 25, 2016 at 8:17 am · Reply

http://www.learncpp.com/cpp-tutorial/01-introduction-to-these-tutorials/
This is a good place to start

abhay agarwal
May 25, 2016 at 12:35 am · Reply

please reply thanks in advance
<code>

```
#include<iostream>
#include<string.h>
using namespace std;
class human
{
public:
string *name;
int *age;
public:
human(string iname,int iage)
{  name = new string;
  age = new int;
```

```
    *name = iname;
    *age= iage;
    cout<<"hello"<<endl;

}
void display()
{

    cout<<"hi i am"<<*name<<"and i am"<<*age<<"years old"<<endl;
}

~human()
{
    cout<<"all memories are released "<<endl;
}
};
int main()
{
    human *anil=new human("anil",18);

    delete anil;
anil->display();
delete anil;
delete anil;

    return 0;
}</code>
```

respected sir

in this code I have declare object dynamicallyand nam and age variable also dynamically ... do I need to use delete keyword in destructor for name and age... sice ihace use delete keyword for object in main function..

Alex
May 25, 2016 at 7:30 pm · Reply

Yes, because name and age are dynamically allocated, you'll need to delete them in the constructor. Because anil is also dynamically allocated, you'll need to delete it as well.

Abhay Agarwal
May 25, 2016 at 9:54 pm · Reply

thank you sir

Jazz
February 16, 2016 at 12:59 pm · Reply

Alex, thank you once again! I've never read such a clear and good composed programing tutorial on english. I have one offtop question.

Would you be so kind to tell me, what kind of C++ knowlege is more applicable for game design, and where to find simple game tasks for C++ newbie? For example, I want to write win32 application that will randomly generate 2D labirinth in console window.

Yaseen
February 16, 2016 at 2:38 pm · Reply

I too wanna know the answer...I am interested in game 2d game development :D

Alex
February 17, 2016 at 1:05 pm · Reply

All of the content in these tutorials are potentially useful for game design. In addition, a good working knowledge of algorithms and data structures will help. You'll also need to learn more about outputting (if you're making a text-based game, maybe using the curses library -- if you're making a graphical game, maybe using the SDL library).

I'm not sure where you can go to find game tasks for newbies though.

Jazz
February 18, 2016 at 2:50 pm · Reply

Alex, I've found such tasks at CodinGame site (I hope you do not take it for advertising). My first sallary at game-dev position, will be yours ;)
cheers

Yaseen
February 1, 2016 at 6:22 pm · Reply

Hi! Alex . I have a question that previous chapters from this chapter are for c or c++ ?

Alex
February 2, 2016 at 12:47 pm · Reply

C++, just building the foundations for object oriented programming (e.g. what is a function, what is a pointer, etc...)

Yaseen
February 2, 2016 at 4:02 pm · Reply

But Alex i wanna learn only simple C not C++ because first i wanna clear procedural programming lessons...

Alex
February 3, 2016 at 1:23 pm · Reply

If you really want to learn C, this is the wrong site. :)
If you really want to learn procedural programming, you can do that with C++. Just start with chapter 0 and keep reading.

There's really no reason to learn C before C++.

Mohammad Etemaddar
November 24, 2015 at 1:21 pm · Reply

Dear Alex, I'll be very excited, If I understand the phrase "never use a small word where a big one will do"
Would you mind give me a hint?

Alex
November 25, 2015 at 1:56 pm · Reply

Aspiring writers are often given the advice to "never use a long word where a short one will do". In other words, favor short words instead of long ones, because short words are easier to read and understand.

However, the C++ language designer seems to favor using lots of long words, like "inheritance, encapsulation, abstraction, and polymorphism". So I was making a joke about this.

Andile
October 13, 2015 at 11:43 am · Reply

Hey Alex, are you able to put up tasks for OOP? It would be of great help as it gets more confusing at this point.

Alex
October 14, 2015 at 12:22 pm · Reply

Yes, I'll add these when I rewrite the lessons.

Banelus
September 4, 2015 at 12:02 pm · Reply

I'd like to know how object oriented programming is associated with making some GUI applications. I found that there are some libraries that allows us to make GUI (like Qt, wxWidgets, WinAPI etc.). If some of these libraries are necessary, could you please recommend the one you find most easy/have the greatest potential?

P.S.
Sorry, but I didn't know where to post it.

Alex
September 4, 2015 at 7:03 pm · Reply

There's no direct relationship between object-oriented programming and GUI libraries (though some GUI library use OOP functionality and principles).

Which GUI library to use is really outside the scope of this tutorial and my expertise. There are a ton of factors involved in picking the right one, including cost, whether you need cross-platform compatibility, etc...

Tim
March 25, 2015 at 3:30 pm · Reply

This might sound like a daft question but does everything in oop other than main() have to be an object (i.e. declared as a class) or is it ok to create objects where appropriate and just use ordinary functions when it is not really necessary to create an object.

I have some code that creates and displays a popup message box which is extremely short lived (1 key press). It seems a bit daft to put it in a class where I must declare a dynamic object then call a method and then delete the object rather than just call a function. That said, I do not want to get into bad habits while I am learning.

It doesn't help that what I am working with is ncurses which is actually a c library.

Thanks

Alex
December 8, 2015 at 1:13 pm · Reply

You can mix both traditional and object oriented programming. It's totally okay to not use OOP for cases in which the added complexity isn't warranted or necessary.

subramanyam
June 11, 2010 at 12:59 am · Reply

Its very nice material..

Bruce
January 16, 2010 at 6:31 am · Reply

I have been rather confused by some of the terminology of OOP, such as 'class', and 'wrapper', etc.; however, it appears as though these things are simply evolutions of the old 'subroutine' construct. Using a subroutine is now handled by the language which acts as a 'traffic controller' making it unnecessary for the programmer to keep up with 'returns' and allowing program flow to continue in any direction. If this is the case then it is not so hard to understand.

Jim
April 6, 2015 at 6:41 pm · Reply

Well, OOP provides another way of organizing not only code, but functions, in a way that is error proof while reusing. It also brings the concept of "members", so that people can organize information in packages, instead of variables.

Pathik
December 25, 2008 at 12:48 pm · Reply

So object oriented programming can only be done with C++...not java or anything other?

Alex
December 29, 2008 at 8:35 pm · Reply

Object oriented programming is available in many of the popular languages today, including Java, C#, Python, Perl, etc...

Ulugbek
March 3, 2016 at 9:26 am · Reply

Hi everyone! I'm new in this site. I'm learning CPP. I have one question. How can I find exercises book for CPP?

Hiu
March 17, 2008 at 5:05 pm · Reply

I'm thrilled after I read this article!! At this moment I'm taking intermediate C++ programming, and soon our teacher will teach us object-oriented programming (hope I'm right). And again, thank you for creating such a great C++ resource for programmers