C++ For Beginners

A Step-by-Step Guide to Learn, in an Easy Way, the Fundamentals of C++ Programming Language with Practical Examples

JULIAN JAMES MCKINNON

C++ for Beginners

A Step-by-Step Guide to Learn, in an Easy Way, the Fundamentals of C++ Programming Language with Practical Examples

Julian James McKinnon

© Copyright 2020 by Julian James McKinnon

All rights reserved.

The material contained herein is presented with the intent of furnishing pertinent and relevant information and knowledge on the topic with the sole purpose of providing entertainment.

The author should thus not be considered an expert on the topic in this

material despite any claims to such expertise, first-hand knowledge and any other reasonable claim to specific knowledge on the material contained herein.

The information presented in this work has been researched to ensure its reasonable accuracy and validity.

Nevertheless, it is advisable to consult with a duly licensed professional in the area pertaining to this topic, or any other covered in this book, in order to ensure the quality and validity of the advice and/or techniques contained in this material.

This is a legally binding statement as deemed so by the Committee of Publishers Association and the American Bar Association in the United States.

Any reproduction, transmission, copying, or otherwise duplication of the material contained in this work are in violation of current copyright legislation.

No physical or digital copies of this work, both total and partial, may not be done without the Publisher’s express written consent.

All additional rights are reserved by the publisher of this work.

The data, facts, and description of events forthwith shall be considered as accurate unless the work is deemed to be a work of fiction.

In any event, the Publisher is exempt from responsibility for any use of the information contained in the present work on the part of the user.

The author and publisher may not be deemed liable, under any circumstances, for the events resulting from the observance of the advice, tips, techniques and any other contents presented herein.

Given the informational and entertainment nature of the content presented in this work, there is no guarantee as to the quality and validity of the information. As such, the contents of this work are deemed universal.

No use of copyrighted material is used in this work.

Any references to other trademarks are done so under fair use and by no means represent an endorsement of such trademarks or their holder.

Table of Contents

Introduction

Congratulations on purchasing ***C++ for Beginners***, and thank you for doing so.

The following chapters will discuss all of the different things that you need to know when it comes to working with the C++ language.

There are a lot of different options that we are able to explore when it comes to handling our work in coding and programming, and many of them are going to come with a ton of benefits that can pull you to them.

But when it is time to work with a simple coding language that has a lot of power and functionality behind it, then it is time to work with the C++ coding language.

This guidebook is going to take some time to look over all of the things that we are able to do with the C++ language, and how great learning this language can really be.

The beginning of this guidebook is going to spend some time taking a look at what the C++ language is all about and some of the benefits of bringing this one to the table, and to your programs, rather than using one of the other many options.

When we have some of the basics down and a good idea of why we would want to use this language, it is time to look a bit more at how we can actually write our codes in this language.

We will spend the next few chapters looking at the right steps to use to help create our first program in this language, and then we can spend a bit more time looking specifically at the syntax of C++ and how that is important in writing out some of the codes that we want to use in the future.

Now it is time to move into some of the fun stuff that we are able to do with this language in particular.

Now that we know how to work with this language, and how to write out a simple code, we can take that a bit further and learn how to work with the C++ variables, and the loops and arrays as well.

We will even take this further and look at a few of the codes that you need to actually create some of your loops and arrays, and look at the basics of how

to add value to your variables so the code can pull them out later.

When we are done with all of this and have a good idea of how to write out some of the simple codes that we have focused on so far, it is time to go a bit further.

While w looked at a few of the operators when it comes to our variables earlier on (and we will discuss how this is so in this guidebook), we will dive more into some of the different types of operators that are present in C++ and why this is so important.

And then we finish out this part by looking at how to create some of our own decision controls statements, which will ensure that our program is able to make the right decisions as it goes, based on the conditions that we set in the code.

Then we are going to jump right into some of the more advanced parts of the C++ coding language that we are able to focus on.

We will look at how to work with the idea of inheritances, which can help us to take full advantage of the OOP part of this language, and then we can dive into some of the coding and more that we need with the idea of polymorphism and getting this to work well for some of our needs along the way as well.

There are a lot of coding languages that we are able to work with, and a lot of them can help you to get your coding done.

But when you want the best language, the one that is easy to learn while bringing on all of the functionality and more that you want in a coding language, then the C++ language is the right one for you to use.

When you are ready to get started with this particular coding language, make sure to check out this guidebook to get started.

Chapter 1: What is C++

When it is time to pick out one of the coding languages that you would like to use, there are a lot of choices out there.

There are languages that are designed to work well for beginners who really have no experience when it comes to coding in the first place.

There are coding languages that are going to work well for those individuals who are looking for a way to create something online to help with a website or an app.

Some are going to work the best with a certain operating system, some work better for gaming, and still others may be used to help out with phone and mobile needs.

There are a lot of choices that you are able to make when it comes to your coding language to get things done.

Knowing what projects you would like to complete and will help you to get your end results are going to make life a lot easier.

In this guidebook, though, we are going to spend some time taking a look at the C++ language and what we are able to do to make this one work for us well.

When it comes to working with a great coding language that comes with a ton of history with it, then you will find that the C++ language is the perfect one to get started with.

In fact, this is going to be considered one of the very first languages out there, one of the first that is set up to help us learn how to code in the first place.

Because this language has been around for a long period of time, there are a lot of serious programmers and coders who are going to learn how to use this and that they understand the basics of this coding language in order to make it work for them.

As we can guess pretty quickly here, C++ is going to be a programming language that is set up to write out some different parts of the codes and the different applications that you are looking to design.

Like many of the other coding languages that you can find on the market

today, C++ is going to be an OOP language or object-oriented language.

What this means to us is that this language is able to revolve all-around classes and objects, rather than the abstracts that come with some other choices.

This makes it easier for us to write out some of the different codes that we want, and you will be able to use this to prevent some common issues that these languages have to deal with.

C++ is going to be a good language in coding to learn because it has been designed to have all of the flexibility that you need to get things done.

There is going to be a lot of speed that comes with this one, which is going to make it an even better coding language to work with.

There are a lot of different ways that we are able to work with C++, including:

1. Prepackaged scripts:

These are going to be really important to you as a beginner because they will help you to practice some of the hacking techniques that we are going to talk about in this guidebook.

These are packaged ahead of time so that you can learn without having to set it all up on your own.

2. Videos games:

C++ is the language that is behind many popular video games that are out now.

While this guidebook is not going to go so in-depth that you will be able to create your own video game when you are done, it is definitely something that you can learn how to do with this language.

3. Web pages:

There are many web pages that will rely on C++ because this language is really easy to manipulate and will make sure that any websites that you create with it are quick and easy to use.

You can even use this language to help you come up with some cool

interactive features.

Popular websites like eBay and Amazon utilize this kind of language.

4. Phone apps:

If you are a fan of using your smartphone and playing with some of the apps that are available, you may find that working with C++ can make it even better.

This language can help you to create some of your own smartphone apps in no time.

A Brief History of C++

The next thing that we need to take a look at here is some of the histories that come with C++.

This is going to be a coding language that has a lot of rich history because it was one of the first coding languages out there.

It started before much was known about computer programming in the first place and long before people even thought about doing this kind of thing as a living.

Think back to the 1979 time period, when things like telephone hacking were starting to be a thing.

That is how far back we have to be in order to learn the beginnings of this coding language.

And you will find that there are not that many other programming languages that are still in use that are going to head back all this far.

Bjarne Stroustrup was working on his thesis for his doctorate, and he decided to work with a programming language that was known as Simula.

This language is one of the first programming languages of the computer age.

However, it was very slow and full of bugs.

Strousup came up with the idea of C with Classes. A programming language that was a lot faster than Simula. C with classes later became Cfront, which sped up the process of creating a language.

However, Cfront was left in the dust when C++ came along, because it added compilers into the language, making it a lot easier, and faster to use than any other project language of the time.

Since then, there have been annotated reference guides and updates to the language to make it better and faster, and even easier to use.

C++ for Dummies is a popular guide for this language.

C++ is actually going to be one of the top languages for coding that is out there today.

This language is going to be the best to use for a lot of industries, and so, rather than taking the time to make up a new language to use, which takes up

a lot of time and effort, they decide to wake C++ and then adapt it to some of the different parts and projects they want to do because of the versatility.

Why is This Language Used By Programmers?

As we mentioned before, there are a number of programming languages out there that we can use, and you may be a bit curious as to why the C++ language is going to be the one that programmers choose instead of some of the other options.

To start with, a lot of programmers like C++ comes with a lot of flexibility and speed.

This is one of the older languages out there, but it is able to keep up with your needs, and it is going to provide us with all of the codes and more that we need in order to get our programs and apps done.

It is also considered a good language for beginners to use because once you learn the basics, you can use these to guide you with other languages in coding.

In addition, if you would like to learn a bit more about how to use a coding language that can handle different projects outside of just working on a computer, then C++ is a good place to start.

It is going to do some wonders with some of the codes that would work on websites, for mobile apps, and so much more.

Of course, some of the options that we are talking about above are just going to be the start of why programmers are going to love working with C++ so much.

Some of the other benefits that will help us to fall in love with this programming language will include:

1. It has a really large library: Since C++ has been around for a good deal of time, it is also going to come with a library that is large.

This is a library that any programmer of this language is able to use, and can make choosing our codes and going with the right functions easier, while also saving time.

You are even able to take this further and create codes on your own if needed, but for a beginner, this library is going to be helpful to get you going on the right foot.

2. It can work well with some of the other languages.

There may be some instances where you need to work with another coding language in addition to C++.

And this is really simple to accomplish when you handle C++.

This ensures that you are able to get all of your coding done in no time, and can make it so much easier to handle a lot of the work that you want to get done, whether your coding needs are simple or more complex.

3. It is able to handle a lot of coding projects.

Often, the other programming languages that you will choose are set up just to handle a minimal number of projects at a time.

For example, using JavaScript as your coding language means that we are just want to create one of our own websites.

But when we use this language, it is able to handle a ton of different projects, not just websites.

It can help us to create new programs, and even work with websites, or even help if you are looking to do something else with your language, the C++ code will be able to help you get it done.

4. Fast and really reliable to work with.

If you have ever had a chance to work with some of the other languages or coding out there, ones that have had some popularity, then there is a chance you know they weren't always as reliable as they promised.

Information may slip through, or they just would not work in the manner that you would need.

If you would like to find something that will actually work the first time and that you can rely on the whole time, then the C++ language is the right one for you.

5. Has the power that you need.

Anyone who decides to work with programming and some coding, and

wants to make sure that the codes they create have the right power to them will enjoy working with the C++ language and all it has to offer.

It is going to have the most power that is necessary and is often seen as a more powerful choice that we will see with some of the other coding languages out there.

6. The C++ language is going to be seen as a low-level language.

This is going to make it easier for a beginner to work with, but can make it really hard for us to do some of the higher-level stuff that we want, and could make us lack some of the features that we are looking for.

7. You will quickly find that the C++ language is going to work with what is known as multi-paradigm programming.

This Paradigm is going to mean that the style of programming is going to be concerned more with using procedure, structure, and logistics in the programming that we do.

C++ is going not just to be the paradigm, but also multi-paradigm, which means that it is going to follow the three main paradigms that are out there, namely that it is object-oriented, imperative, and generic.

8. If you are looking for a coding language that will provide you with all of the performance to handle even your bigger projects, and a good amount of efficient memory in the process, then the C++ language is going to be the right one for you.
9. C++ is going to be a language that is able to work with other languages, and it is automatically compatible with the C language.

If you have worked on the C language in the past and would like to combine these together, then you will find that the C++ s completely amendable to this, and it is worth our time to learn how to make this happen.

10. There are a number of other features that we will enjoy when it comes to working with the C++ language, based on our overall goals when it comes to using this for coding.

It will allow us to reuse some of the codes, such as is necessary for inheritances, and it is going to accept the process of polymorphism as well.

There is also a good deal of portability that will happen when we use this kind of language.

The portability that comes in C++ is going to allow us to develop some of the programming that we need, no matter what kind of hardware we want to use.

It is also going to allow us to move the development of all our programming form one of the platforms over to another if this is ever a concern for us.

These are just a few of the reasons that programmers like to work with the C++ language.

It has all of the compatibility, power, features, and more that we are looking for when it comes to handling the coding projects that we have, and it can really ensure that we get the work done.

Sometimes it is criticized for being a bit tougher to learn how to use than some of the other options like Python, but it has all of the parts that you need and can get everything done in no time at all.

As we can see here, there are a lot of benefits that we get to enjoy when it is time to work with the C++ language.

It may not be the first language that is going to show up when you are trying to learn a new coding language, but it is one of the best, and it is going to be a great option to ensure that we are able to get the results that we want in the process.

Whether you are looking for a language that is able to handle a lot of powerful programs that you want to create, like games and more, or you just want to be able to add something new to your toolbelt of coding languages that you want to learn along the way, you will find that the C++ language is going to be the right choice for your needs.

Now, we do need to remember that even though the C++ language has a whole host of benefits to work with as we discussed above, there are going to be a number of drawbacks to working with this language, which is why not every programmer out there will choose this one for their needs.

The first negative here is that there isn’t any security with this language.

In a world that is very connected, and exploitation can happen at any turn, the lack of security is going to be a big turn off for a lot of programmers.

A few of the other negatives that we are going to need to consider when it comes to using the C++ language include:

1. It is going to be hard to work with when we want to combine it or use it in a high-level program, especially one that is large.
2. It is often going to be used just for some platforms that are application-specific.
3. It is often going to be used just for one particular platform or operating system.

   It could be switched over to something else if we would like, but the library is going to lock on the one and will struggle when moved in some cases.

4. When we try to work with C++ for some of our web applications, the coding can be a bit complex to handle, and often there will be a lot of difficulties when it is time to debug it.
5. This language is not going to support the process of garbage collection.
6. C++ is not going to be secure because it is going to come with a few different parts that prevent this, most notably the global variable, friend function, and the pointer.
7. And there will also not be any support for the threads that are built into all of this.

While we are often able to work around some of the negatives that are going to show up with these issues, we have to remember about them and think about whether these are important to the work that we want to do in our coding. Weighing the positives and the negatives are the best way to make sure that we will end up with the right coding language, such as C++, for all of our needs.

Chapter 2: The Steps to Creating Your First Program

Now that we know a few of the basics that come with coding in C++ and why so many programmers want to spend their time learning how to make this work, it is time to take everything to the next step as well.

In this chapter, we need to take a look at some of the steps that we need to take in order to write out our own codes.

To do this, we need to work on the basics, and then you can take these on to write some of your own codes at a later time.

The first part of the process, before we are able to write out ay codes, is to work on the C++ environment.

The environment is simply the place where we are able to write out some of our codes and then set it up to execute, which is why we need to make sure that this is ready to go and that we use the right one.

The good news to remember here is that there are many environments for C++ out there, though the best one is usually found on the main website for C++.

Once the environment for this coding language is downloaded and ready to go, it is time to learn the right steps to writing our first code.

Our goal in this chapter is going to write out a code that shows the phrase "Try this" on the screen of our computer.

This is only going to take a few lines to accomplish, but it does include a lot of the syntax that we need in our codes to see the best results that we want as well.

Just because it is simple doesn't mean that it won't work the way that we want.

With that in mind, the code that we need to use to make this one behave well includes

```
#include <iostream>

using namespace std;

int main ()
```

```
{
        court << “Use This One!”;
        return 0;
}
```

You have a few options available when it comes to writing this out.

You can choose to write this out in your compiler, which will be available in your environment, or you can choose to write it out and save it to your computer.

The second option is sometimes nice because then you would have the code saved and cold copy and paste it any time that you would need it in your code.

Either way, you should carefully consider the type of text editor that you want to work with.

Most of them are going to be device-specific, so you will need to look for the one that goes with your particular computer.

Some of the options that you can go with include Windows Notepad, vlm, vl, Brief, and EMACS.

If you would like to have an editor that is able to go on more than one platform, then the vlm and vl options are the best ones.

When you are working on writing some codes in C++, you need to make sure that they are written out in the text editor.

This is going to provide you with a good rough draft on the program, and then, when it is done, it can be run by the compiler.

This makes life easier for you because you get the option of checking the code before running it, and ensuring that there are no longer any errors found inside of it.

The C++ Compilers

While we are here, we need to take a look at the C++ compiler and see what job it is able to do for us here.

Just like how we talked about with the environment and the text editors, there are going to be a large number of compilers that you can choose to work with.

The problem with all of this, though, is that while you do get a number of choices to work with on these, you will quickly notice that a lot of the compilers are going to be really expensive to use.

The main reason for this is that they are going to be saved for, or have special features for, elite hackers.

These are the individuals who have been able to master the lower level compilers that are out there and who are looking for the steps that they need to make it to the next level.

A lot of the C++ compilers are going to cater to this group but at a cost, of course.

The good news here, though, is that all hope is not lost.

There are some great C++ compilers out there that are safe and secure, while also being free.

You just have to be careful and watch what you are doing to ensure that you are finding the ones that are the best and will still have all of the features that we want to work with.

One of the different compilers that can help us with our C++ coding, and will have some features that are good for beginners is known as GNU.

It is going to be the best one to use with the Linux system, and if this is the operating system that you want to work with, then this is already there.

To check out whether the GNU compiler is already found on your computer, you just need to use the code below:

$ g++ -v

If the compiler is installed, then you should see this message on the screen:

Using built-in specs.

Target: i386-redhat-linux

Configured with: ../configure –prefix=/usr …...

Thread model: posix

gcc version 4.1.2 20080704 (Red Hat 4.1.2-46)

If you typed in the code above and you did not get that message to come up on the screen, it means that you either did not download the compiler the proper way or you do not have it on your computer in the first place.

Some of the Basic Syntax to Understand

Before we end out this chapter, we need to take a moment to look at some of the basic syntaxes that are going to come with the C++ language.

This is going to be important to ensure that, when it is time to move on to writing our own codes later, we are actually able to get it all done.

The C++ language is one that we are able to define as a program that will use lots of objects, objects that are then able to help us keep things organized, and will allow our code to communicate as well as invoke some of the other methods that are there.

There are four main parts that are going to show up with this kind of language and will be seen as an important part of the syntax that we are working with as well. these four main parts include:

1. The classes:

These are important because they are the tools of organization in the C++ language.

They can be seen simply as little boxes that are going to hold onto objects, and then can sort out some of the objects that are similar to one another.

You are able to give any label that you want to the class, but it is often good practice in coding to place objects that are similar to the class in some manner together.

2. The objects:

The objects are next on the list, and they are the parts of the code that will come with states and behaviors.

These could be things like shapes, textures, colors, and more.

You will be able to classify these objects into the classes they belong to based on how similar they are to the other objects in that class.

So, if you have a class that was labeled dogs, you could add in any objects that talked about dogs into it.

3. The methods:

The methods are next, and this is going to be one of the terms that we are able to use for coding behavior.

There are as many or as few of these methods as you feel comfortable with using, and they are going to be able to manipulate some of the data and actions and how they will play out on your code.

Without the right method in place, then the program is going to b at a loss for how you want it to behave when you execute it.

4. The instant variable:

These are going to be the parts of the code that will refer back to some of the individual objects that you want to work with.

Each one is going to be classified using a unique set of variables that are similar to the fingerprints of the object.

You will be able to use some of the values that are there to help you assign the correct type of variable to your object when it is first created and ready to go.

And that is the basics of what you need to know when it is time to work with the basic syntax of the C++ language.

We even took the time here to look at how to write out one of the first codes that you need to utilize when it comes to handling the C++ language so that you can get a good idea of what you are supposed to do here, and how it will behave.

To help you get more prepared on what needs to happen when you are working with the C++ code, make sure to take that code above and give it a try.

This will make it easier for you to go through and actually get more familiar with how this language works, and what you are able to do in order to see the best results along the way.

Chapter 3: The C++ Syntax

In the previous chapter, we spent a bit of time getting some familiarity with what we are able to do with some of our coding in C++.

You can see that writing out a code is not supposed to be that complicated when it comes to handling this language as a whole, but there are a lot more complicated things that we are able to do with this language, and we will get into that a bit more as we progress through this guidebook some more.

However, we need to take things in a different direction now.

This chapter is going to spend some more time looking at how the syntax of the C++ language is meant to work, and some of the different parts that need to show up so that we are able to actually make this language behave in the manner that we want.

As we go through this chapter, take some notes on what we are doing here, and then be on the lookout to see when these parts show up and how often they are actually used in some of the codings that we want to do.

Remember that earlier we took a look at a good example of how we are able to write out code I the C++ language, and even some of the main parts that are going to show up in the code as well.

Now that this is all done, it is time to dive a bit more into some of the things that we can do with the C++ syntax.

The syntax of any language, whether we are talking about C++ or not, is going to be so important because it will ensure that the code turns into something that our compiler is able to understand and execute in the long run.

With this in mind, it is time for us to learn more about the syntax of the C++ language so that we can use it well.

The Comments in C++

The first part of the syntax that we are going to take a look at in the C++ language is the comments.

These are going to be a nice thing to learn about because they will ensure that we are able to leave a little message or a note in the code that we are writing, without it affecting how the code is going to behave.

These comments are going to help us or others who take a look at the code, get a better understanding of what you would like to accomplish with this code.

As a beginner, you will find that a lot of codes that will include these kinds of comments so that you will have a better idea of what is going on there.

You will find that these comments are going to help to guide you with what you are hoping to get done in the code, and we are technically able to add in as many of these as we want into our codes.

The rule of thumb here, though, is that you want to limit the comments to just the ones that are necessary so that you don't make the codes a mess.

To make the comments in this language, we need to use the (//) symbol ahead of the comment.

This is going to tell the compiler that you are writing out one of these comments, and it should just skip over this part and not include it int the execution that you work with here.

You can add these into any part of the code that you would like, and the compiler will know what is going on.

It is also possible for us to write out a comment that takes up more than one line, or is a little bit longer even in this language.

This is something that is a bit hard to do in other languages but is unique to the C++ language.

When you want to go through and have a code that takes on more than one line, you would add in the /* to the beginning of the comment, and then whenever it ends, you would use the */ symbol.

When the compiler sees this, then it is going to know that it should proceed through the code without reading any of this at all.

The Different Types of Data

Each coding language that we want to spend time on is going to spend their effort looking at the idea of the types of data.

These are going to help push the code forward and can make it easier for you to actually store the information that you would like along the way.

And C++ is going to value these types of data just as much as all the other languages as well.

You will find that when coding in the C++ language, the data types and the variety of them will be just as important as it is in any other language.

Often you will spend time working with the data type that is known as variables, for example, which are going to be spots that are reserved in the memory of your computer, and will make sure that specific values are safe, and can be pulled out when it is necessary.

There are a lot of other types of data points that we need to work with as well, and some of these include:

1. The valueless
2. Double floating-point
3. Wide character
4. Floating-point
5. Integer
6. Character
7. Boolean

Starting New Lines At the Right Time in C++

The spacing and the lines that you use in this language will be really important to the amount of success that you are going to see with it.

If you do not space things out in the proper manner, there could be some trouble with reading and understanding the code that you write, and this can add to more trouble later on.

When you decide to write out any of the codes that are needed in your program for C++, it is important that you go through and use the right line spacing to make it work.

It is not necessarily a good thing for us to go through and write out the whole code on one single line, without pauses or any breaks to it at all.

This is technically fine to do because the compiler will be able to read through it whether you do this or use another method of separating things out.

But when an actual person takes a look at it and tries to figure out what is showing up there, they will be confused because it is such a mess.

To see how this works and what we are talking about here, consider the following example:

```
{
        cout << Try This;>> endl; << Today I ate Pizza and did Math;>> endl;
<<6=(7-1);>> endl; << That is what I learned today;>> endl;
        return 0
}
```

This line is technically right.

You would be able to put it into the compiler, and it will read through all of this just fine.

However, it is really hard to read through this, especially if you are brand-new to the idea of coding.

This particular one would not be that bad, but if you get to a longer code, it would be almost impossible to figure out what is going on or even to catch mistakes that you may have made.

There is another way that you can write out the code.

It will mean the exact same thing, and the compiler will read it the same way, but it is better because other programmers, as well as yourself, will be able to read through the code a little bit better. a better way to write out the code includes the following:

```
{
        cout << Try This;>> endl;
        <<Today I ate Pizza and did Math;>> endl;
        <<6=(7-1);>> endl;
        << That is what I learned today;>> endl;
        return 0
}
```

As you can see, this is much easier to read through.

It says the exact same thing as the first one, but it looks a lot better and it easier to read.

Good coding practices will ask you to write like on the second example, rather than the first one, although your compiler will accept both.

The C++ Functions

We can't get too far in our discussion about how to work with C++, or any coding language for that matter, without first taking a closer look at how the functions are going to behave for us.

When we work with the functions, this simply means that we are focusing some attention on a group of statements that are supposed to work together to help us get a specific task done in the process.

You will find that each of the programs and applications that you use C++ to create will have at least one function in it, which is going to be known simply as the main() function here.

But it is common, especially when you are working with some of the longer codes and programs, to have a bunch of functions that will show up in your code.

There are a few ways that we can take these functions and make them work well.

First, we can take our code and divide it up so that we have several functions in control, based on what the code is supposed to do.

How you go through and divide up the code is going to depend on what you are trying to do with that code overall.

But the most logical choice that a lot of programmers are going to focus on is to divide up all of the functions so that they are able to handle some of their own tasks.

To get the function to work at all, though, we need to declare that function.

A function declaration is going to tell the compiler about the functions name, the parameters that we want to have around the function, and the return type that we want to get back.

However, the definition of a function is going to be provided in the body of any function that we want to create.

The standard library that we see with this language will provide us with a number of functions that are already usable and that we can call up.

For example, one of these is the strcat() function, and it can be used in order to concatenate two strings, and then you can use the function of memcpy() to

help us take the location of memory from one spot to another one.

These are just a few of the functions that you can work with and they will work in a similar way.

So, let's take a look at how the functions in C++ will work.

You can define it with the following:

```
Return_type function_name (parameter list ) {
        Body of the function
}
```

The definition of the functions that we have here are going to consist of a header for the function, as well as its body.

Some of the different function parts that we need to look at will include the following:

1. The return type:

   In some cases, when we are using a function, it is going to return a value of some sort to us.

   When this is going to happen, we are going to work with the return_type in order to get something to actually come back to us.

   Some of our functions will be able to do the operations that you would like here without returning value to you.

   With the syntax that we wrote out above, you would get a value returned, though the value is going to be "void."

2. Function name:

   Then we are able to work with the function name.

   This is going to be the name that you can give to your function with a parameter when you are ready.

   You will also end up getting what is known as the function signature as well.

3. Parameters:

The parameters are simply going to be a kind of placeholder to work with.

When you are ready to invoke one of your functions, you will simply pass value over to one of the parameters.

This value is going to be referred back to as the argument or the actual parameter.

The parameter list will then be able to refer to the order, the number, and the type of parameters that go to the function.

It is possible, in some cases, to work with a function that does not have these parameters in place at all.

4. Function body:

And finally, we need to take a look at the function body.

This is going to contain a good collection of statements that are responsible for telling us what the function is able to do.

While we are here, it is also possible that we can go through this and work with something known as function overloading.

What this means is that the C++ language is going to allow us to specify two or more definitions to the name of a function, or to one of our operators, all within the same score.

This is going to be known as operator overloading or function overloading based on the one that we do this process to.

An overloaded declaration is simply going to be one that is declared using the same name as the declaration that we declared earlier on, as long as it is in the same scope.

The exception to this is that both of the declarations that we have here are going to be from different arguments and will have different definitions from one another, and different implementations.

When we call up one of the operators or the functions that have been overloaded, the compiler will be able to take a look at all of the definitions and then determine the one that is the best to use.

It does this with a comparison of the different types of arguments that you have used in order to call up the operator or the function, and it can also work with the types of parameters that are specified in the definitions.

This is a big process that the compiler is going to need to work with.

The good news here, though, is that we will be able to see the right results each time, as long as we are able to get the program set up in the right manner.

The process of selecting the most appropriate function that has been overloaded is going to be overload resolution.

As we mentioned, it is possible for us to go through and have more than one definition for the same function even when we are using a scope that is the same.

The definition of our function has to be different from one another through either by the type or how many arguments that will be found in one of our lists for arguments.

Remember that it is not allowed here to overload the declaration of our function if it is only going to differ by the return type.

It is pretty easy to work with this overloading of functions, even though it may sound a little bit difficult when we are getting started.

A good example of the coding that we can use to make this one happen is going to be below with the print() function being used to help us print out all of the different types of data.

```
#include <iostream>
using namespace std;
class printData {
   public:
      void print(int i) {
        cout << "Printing int: " << i << endl;
      }
      void print(double  f) {
```

```
        cout << "Printing float: " << f << endl;
      }
      void print(char* c) {
        cout << "Printing character: " << c << endl;
      }
}
int main(void) {
   printData pd;
   // Call print to print integer
   pd.print(5);
   // Call print to print float
   pd.print(500.263);
   // Call print to print character
   pd.print("Hello C++");
   return 0;
}
```

When we are done compiling his one, and we can go through and execute it well, it is going to give us some results about the types of data that we added into it.

We can do a similar process when it comes to working with operator overloading in C++.

We are able to go through and overload or redefine some of the operators that the C++ language has built-in.

Thus, the program is going to be able to use operators with some of their user-defined types as well if this is their choice.

To keep it simple here, the overloaded operators are going to be functions that have special names.

Usually, this is going to include the keyword “operator,” and then this is going to be followed by the symbol based on the operator that you would like

to define here.

Just like with any of the other functions that you are using, you will find that the operator that you are trying to overload is going to come with both the return type and our parameter list.

The Types of Modifiers

The next thing that we need to take a look at here is the modifier types.

When you are ready to work with some of the code that is found inside of this specific language, you will find that char, int, and double are all allowed to come with a modifier that will come ahead of them.

The modifier is going to be used in a manner to help alter, in one way or another, the meaning of that base type, which is going to help us better get it to fit with the current situation we are dealing with based on the kind of program that we want to create here.

You will quickly notice that there are four options when it comes to modifiers of data that work in C++.

These are going to include short, long, unsigned, and signed.

These four modifiers are going to be applicable to any of the types of integer bases that are there.

You are able, for example, to take the unsigned and the signed and then use it with the char data type, and then the length can only be used on the double.

C++ will also allow you to use a shorthand notation for declaring the long, short, and unsigned integers.

This means that you are able to use those words without having to add in the word “int” because this is going to be implied.

There are also different types of qualifiers in order to get things to work on your C++ code.

The following types of qualifiers that you will be able to use in order to provide some more information that concerns our variables will include:

- Const:

Objects that have the “const” will not have the ability to be changed by the program while you are executing it.

- Volatile:

The modifier of volatility will tell the compiler that you are able to change the value, but these changes may not be specified in an explicit

manner by the program.

- Restrict:

We will find that one of the pointers can restrict things and can be qualified in that manner.

This means that only the object that is pointed at during that time can be the one that we access.

How to Handle Exceptions

One thing that is kind of neat about what we can do with the C++ language is that it is set up to handle things like exceptions.

This is a bit more complicated, but you will be amazed at some of the different codes that we are able to do with it.

An exception, as we will use it here, is going to be a problem that will show up when you are ready to execute your program, which means that it is not going to work properly.

When we look at the exceptions in C++, you will find that this is what will happen in response to an exceptional circumstance in the coding, one that is often going to show up when we execute and run the program.

The most common type of exception that we are going to find within our coding and our programs is when someone tries to divide by zero.

Exceptions are going to be a useful component to have in your code because they will help us to transfer some of the control that is there in one part of the program and make sure that it is moved over to one of the other parts.

The C++ exception handling is going to be built on some of the main keywords, three of them to be exact, including a try, throw, and catch.

To get a better understanding of why these are important and what we are able to do with them, let's look at what they mean below:

1. Throw:

The first of these keywords is going to be known as a throw.

The program is going to throw out an exception when it sees a problem that shows up.

This is going to happen automatically when we work with the throw keyword.

2. The catch:

A program is going to be able to catch on to one of the exceptions with the help of an exception handler being in place in the program so that it can actually handle some of the problems on its own.

If you use the catch keyword, it is then going to take the time to tell your program that you would like it to catch that exception.

3. Try:

When we are ready to bring out the try keyword, it is going to be done to help us look for and identify the blocks of code where a particular exception is going to be activated.

When you go along with this particular keyword, it is going to also have, at a minimum, one catch block to help make sure that it is able to work.

We are then going to make the assumption that your current block is able to raise up an exception, that a method will then be able to catch the exception using a combination of the catch and try keywords.

These are then going to be placed in the code so that it is able to get the exception and handle it for you in the process.

This is not a guarantee that an exception is going to happen, but if there is the potential, adding these in will ensure that the code keeps behaving in the manner that it should.

As we can see here, there are a lot of different parts that are going to come together to help us to write out the basic syntax of our C++ code.

You will be able to use as many of these parts as you would like along the way to make sure that the coding is going to work well, even though some of the basic codes that you start with may only use one or two of the parts to actually make them work.

Learning these basics and the right way to write the codes will make all of the difference as we progress with this guidebook and beyond with this coding language.

Chapter 4: The C++ Variables

We talked about the variables a bit as we have gone through this guidebook, but now it is time for us to get into a bit more when it comes to the details of what these are and why they are so important.

Variables are going to show up in pretty much any of the coding languages that we want to work with, so learning them now is going to be so important to helping us to see some results in the process.

They are basically going to hold onto some of the values that we want, storing them in the memory of our chosen computer so that the compiler is able to pull them out and use them when the code has been executed.

One thing that is important when we are working on all of our coding in C++ is the variables.

These are going to be important because they will help us make sure that the various aspects of our codes will match together, and that the program is actually able to find these parts and pull them out when it is time to use them.

When you work with some of these different variables in your own code, you are basically working to save a spot in the storage of your own computer.

This is going to make it easier for you to bring them up when the code needs to execute them later on.

All of the variables that you will work with can be attached to different types, and then these are the types that are going to help determine the size and the layout of the memory that the variable is going to rely on.

It is also going to be useful when it is time to set up a range of values that you can use to store on that specific spot in the memory.

With that said, you are going to find that naming out the variables is going to be simple.

In fact, it is the same process that you can use whether you are naming out one of your other identifiers or not.

You are able to work with the letters, numbers, or the underscore to help name your variables.

Make sure that you are not starting this out with a number, and all of the

letters that you use are going to be case sensitive.

Outside of those few rules, you are able to pick out any name that you would like to use on the identifier.

One thing to keep in mind with the variables in C++ is that there are many types of variables that you can choose from.

Some of the most popular of these variables will include:

- Wchart_t:

This will be known as the variable for wide-character types

- Double:

This is a floating value that is going to have double precision

- Void:

This will be the variable that shows up to represent that there isn't a type being used.

- Int:

This is an integer

- Float:

This will be the floating-point, and it will only have a single precision.

- Bool:

This is a variable that will return results that are true or false.

- Char:

This is the representation of one byte in the memory and will be a type of integer.

It is also within your power in this kind of language to define some of the other types of variables that you need to use in your coding.

These variables are going to include a lot of options to help you out, including the classes, references, arrays, line pointers, enumerations, and data

structures.

How to Define the Variables

Now that we know a bit more about these variables and what they are all about, it is time for us to actually look at some of the steps that we need to follow in order to actually define some of our own variables.

These definitions are important because they will alert the compiler to how much storage we want it to use, and where we would like to store and then create the variables.

Doing this definition is going to help us to go through and specify the type of data that we want to work with as well.

We could type out something like “type variable_list” and then you could choose from one of the variable types that we listed out above and write it out in the same manner if you need it.

These are going to end up telling the compiler to create the type of carriable that you have gone through and listed, and then get the code to work well.

This is going to sound a bit complicated when we try to list it out and how to make it work, but it is time for us to look at a coding example so that we can see this more in action in C++:

```
#include <iostream>
using namespace std;
int. main ()
int j=10;
int k=5;
int l=j+k
{
        cout <<l>> endl;
        return 0
}
```

You should get the answer 15

You can also go through and define as well as declare the variables that are in

your program, but that is something that we can work on after getting some more practice in this language.

It is a good idea to bring up the compiler and try out the code above.

This helps you to get more familiar with writing codes and can help you to start recognizing when you will see these variables.

While we are on the process, we also need to take some time to assign some kind of value over to our variable as well.

This is going to be important, or we are just reserving an empty spot in the memory of our computer.

On its own, the variable is not going to do that much and will just store some space on the memory of your computer for all of the work that we are trying to do here.

It is your job and actually adds in the value that is necessary so that the memory will be able to hold onto something as well.

The value that comes with a specific variable is going to be stored in the exact same spot of the memory as your variable.

You are also able to take this a bit further because you can store more than one value to the same variable in your program if you would like.

You can technically go through and add as many of these values to the same variable as you like, as long as it is done in the right manner.

However, it is best for us to keep the ratio of our value to a variable just at one or two because this will keep the code as simple to work with as possible.

In order to assign a new value over to one of your variables, all that we need to work with is the equal sign.

This is going to be there to tell the compiler that your variable, or the space that you have been reserving in the memory of your computer, is going to belong with that specific value when the code runs.

When the variable is called up later, and the program is executing, it is going to bring up that particular value that you want.

As you can see here, the variables in C++ will not be that difficult to learn about, but it is so important to learn how to do these to ensure that your code is going to work in the manner that you want and to ensure that all of the

values that you would like to use in your code are going to actually show up when they need to work at the execution of the code.

There are a lot of times when you will want to bring in the variables that work with C++.

And they are going to be useful for making sure that you can actually store all of the values that you would like inside of the code.

Getting this all set up and ready to go is going to be important here, so making sure that we take the time to work with our variables in the right manner is going to be so important to see the results that we want in the process as well.

Chapter 5: The Arrays and Loops

Now it is time for us to look a bit more about some of the codings that we are able to do when we want to write out programs in C++.

We are able to actually create some amazing different codes when we are working with arrays and loops, so it is important that we learn the best steps in order to get this all under control as we need.

In this chapter, in particular, we are going to learn more about two of the processes that we can do when writing out some of our own codes in C++, including the arrays and the loops, what these mean, and what we are able to do with them in our codes.

The Arrays

The first part that we are going to take a look at is the arrays.

Arrays are one of the data structures that are available in C++ that are going to help us to store elements that are pretty much the same type and are going to come in at the same fixed size.

These are basically going to be a collection of the same type of variable that we want to work with.

But instead of going through and using some of the individual variables to get it all done, you will need to work with one array, such as an array of numbers.

To make these arrays work, you would need to stick with the numbers that go from 0 to 99, and then you would be able to access each one using the index of the array.

It is possible to work with arrays that are not just about numbers, though.

We can stick with ones that are for characters and words as well if we would like.

The arrays are going to be similar to some of the variables that we talked about before in that they are going to be locations in the memory of our computer.

But they will be a bit more continuous than what the variables can provide.

The lowest is going to be known is the first element, while the highest element is going to be found as the last one.

You do get to pick from a few different options when it is time to work with these arrays.

You can go through and initialize them one by one, or you can go there and work with the sing statement.

A good example of doing this is going to be the following:

```
double balance [5]= {1000.0, 2.0, 3.4, 17.0, 50.0};
```

The numbers that are found in the bracket that we wrote should never end up higher than the number of elements that you plan to use.

This means that if you place a five in the brackets, as we did above, and then place six numbers into the array because there will not be enough space inside the memory to do this.

However, it is possible not to specify the size of the array.

This is helpful if you are not sure how big the array will be ahead of time so that you can add in as many numbers as are needed.

You would write it out like the following:

double balance [] {1000.0, 2.0, 3.4, 17.0, 50.0};

This one above is going to provide us with the same kind of array that we see with the previous example.

The only difference is that you did not start out the array by specifying the size that it was supposed to be.

This can be useful if you are not sure about the size ahead of time.

The program is going to handle all of this for us.

Now that we have taken a look at what these arrays are able to do and how we can write them, it is time for us to learn how we would take all of this information in order to make the array work inside of any programs you write out.

This program is going to seem a bit more advanced in some instances compared to what we have done in the previous chapters.

The good news is that the example of doing these arrays that we will have below will help us to see how this works and learn how we can easily add in the arrays without making them that complicated at all.

The example that we are going to spend some time on include:

```
#include <iostream>
using namespace std;
#include <iomanip>
using std::setw;
int main ()
```

```
{
        int n[ 10 ];  //n is an array of ten integers
              // initialize elements of array n to 0
        for ( int i=0; I <10; i++)
        {
        n[i] =i+100; // set ekement at location I to i+ 100
        }
        cout << element << setw(13) << value<< endl:
        //output each array element's value
        for (int j=0; j<10; j++)
        {
        cout << setw(7)<< j << setw(13) <<n[j] << endl;
        }
        return 0
}
```

To better understand what is going on here, we need to open up the C++ compiler and then write out this code.

This will give us a bit of practice when we are going through this language to see how it is going to work for us and can make it easier to learn some of the parts of the array.

This program is going to be usable with the array in order to use the setw() function so that you are able to format the output that you see.

This is, of course, just one of the examples of how we are able to work with the arrays and get them to behave in a manner that works for our codes.

How to Work with Loops

Now that we have had a chance to take a look at some of the arrays, we need to look at how we can create loops in C++.

These loops are going to be helpful because they save a lot of time when we want to write out our codes.

If you need to get your code to repeat a single part a number of times, whether it is just a few or a large amount, then you do not want to waste all of your time writing out that many lines of code unless you need to.

This can take up a lot of time, gets tedious, and can make the code harder to read through.

With a loop, you will be able to take some of the code that you already wrote out, and then, after writing it out just one time, you will be able to set it up to repeat itself a number of times.

The number of times the loop will repeat is going to be based on how long it takes until the condition is met.

Make sure that when you write out the loop you actually add in the condition, though.

This is easy to forget when we are writing codes, and then the loop gets stuck and will not be able to stop it unless you end the program.

There are actually a few types of loops that we can work with, and it depends on what you would like the code to do in concerns with them.

The main types of loops that we are able to work with include:

The While Loop

The first loop we can look at here is known as the while loop.

The while loop is the choice that is going to continue on through our loop and will repeat the necessary lines, as long as the conditions you set are met.

The program is going to read on through the information and then test for that condition each time that it does the loop.

Then the loop will restart again if it finds that the conditions were met.

Once it gets to the point where the condition is no longer true, the loop is going to stop and will head on to the next part of the code if there is more to

your program.

With that in mind, the while loop is going to be pretty simple to work with.

To make a simple while loop, you really just need a few lines of code to get it done.

The syntax that we are able to utilize in order to work on the while loop is found below:

```
While(condition)
{
        Statement(s):
}
```

The For Loop

Now it is time to move on to the second type of loop that we can work with is known as the for a loop.

This for loop is the one that only repeats itself after first testing that the conditions are true.

With the while loop that we did above, the loop will go through, and then test the conditions.

With the loop, the conditions are tested first, and then the loop will proceed.

This means that based on the input that is given to the computer or the program, it is possible that the program will test the condition, it will be seen as false, and the for loop will not happen even once inside of the program.

If you are doing a program where the loop has to happen a minimum of one time, then the while loop is the right one to choose.

If it isn't too important, or the code will be fine in the case of the loop not going if the conditions are not met, then the for loop can be a good one.

If you are working on your program and you determine that the for loop is the right one for you, then you will need to go through and write out some code to do this.

It is similar to the while loop that we did above, which is going to make it

easier to handle overall.

The code that you need to use to make the for loop behave in the right manner includes:

```
For (iit: conditions: increment)
{
    Statement(s);
}
```

The Do While Loop

The third type of loop is going to be a unique one that works well in the C++ language but is not going to be discussed as much with some of the other coding languages that you are able to handle along the way.

This is going to make it a lot of fun to work with, so that is why we will spend some time looking at this here.

The do while loop is a bit different, though you will notice that it has some similarities to the while statement that you are working with.

However, it is going to be different in that it is able to wait to check on the condition until you reach the end of the statement, rather than doing it at the beginning.

This makes it work slightly differently in the program that you are writing.

The syntax that we are able to use in order to write out the do while loop includes:

```
Do
    {
    Statement(s):
}while (condition):
```

The Nested Loops

And next on the list that we need to focus on here is going to be known as the

nested loop.

These are going to be a bit more complicated than some of the other loops that we are doing here, but it is going to be useful and can help us to get a lot of things done in a short amount of time.

Basically, the nested loop is going to be when we have one loop running inside of another loop, and both of these loops will continue to run, going at the same time until their conditions are met so they can end.

This can technically be something that results in a continuous loop of loops if we would like.

This can be confusing, but the syntax is going to be easier to work with as you go through and learn a bit more about coding in the C++ language.

In fact, these loops are pretty common to work with because they are simple and easy to handle, and they will help you to cut down on how many lines of code you have to write.

Even though these nested loops are going to be a bit more complicated to write out and work with, they are not that bad, and a good example of the syntax that we are able to work within order to write out some of our own nested loops includes:

```
while (condition)
{
        while (condition)
        {
        statement(s)
        }
        statement(s) // you can put more statements
}
```

Breaking Down the Importance of Loops and Arrays

Now that we have had a chance to go through and learn a bit more about the arrays and the loops, what they are used for, and some of the coding that can allow us to get them done, it is likely that you are curious as to why both of these topics are going to be so important in some of the work that we are doing in our C++ language.

Why would we want to learn how to use some types of coding that seem really complicated and hard to work in the first place?

Couldn't we just skip over this and find something that is a bit easier to handle and still get the codes done?

Even if you just want to focus on some of the basics that come with any language, at some point, you will want to expand out to some of the things that are a bit more complex in order to get your codes to behave in the right manner and to really make some good and strong programs in the process.

You may like that you are able to handle some really simple tasks and can write out some simple codes in no time, but you are not going to get too far working with those basics without progressing to some of the harder stuff.

The basics will provide you with some of the foundations that you need.

And we spent our time on those in the first few chapters that are found in this guidebook.

But now it is time for us to move on and take a closer look at some of the other topics that are going to come with C++.

This helps us to learn and grow and keeps the coding from getting boring.

Even though we can agree that the loops and arrays are a bit more advanced than what you may want to do when you first start, they are going to help us to really get our codes to be as powerful as possible and will prepare us for some of the more complicated options that we need to handle as we go through the rest of this guidebook.

By all means, if you still need some more time with the basics, go back to the earlier chapters that are in this guidebook and practice some more.

We do not want to progress until we are ready.

But once you are ready to challenge yourself, you will find that these are going to help out so much.

And the next step is always going to be important when it comes to coding, as you want to make sure that you are able to progress and get so much better at the coding that you are doing.

You want to make sure that you don't stay stagnant, and that you are able to actually progress to the next level with your programming going from the beginner level up.

The neat thing here is that there are actually quite a few things that we are able to do when it comes to handling these arrays and the loop statements.

For example, there are a lot of times when we are going to bring out the loop statements when we are working on robots and video games.

If you want to work on a code or a program that is able to handle some of the requests of the user without the programmer planning for everything along the way ahead of time, then the loop statements are going to be really great to work with.

Then there are the arrays, and they can be used in many situations as well.

The arrays are often used as part of the web pages for shopping websites because you are able to easily add in a list of variables and not have to do that much with them past the initial work during setup.

You will find that in a business or even an individual setting, there are a lot of times when you would need to create your own website.

Whether you are hoping to write that website from scratch for yourself or the client or you would like to write it for personal use, then you will find that the arrays will help to get this done in an efficient and easy manner.

Even if you decide to make things easier and work with a website that is premade, knowing how to work with these arrays and how to read some of the codes as we discussed in this guidebook, it will be easier to understand what is going on in the code and even make some of the changes that are needed to personalize that website for your needs.

There are also going to be many times, in addition to some of the options that we talked about earlier when the loops will show up in your coding and can save you time, get certain parts of the code to repeat a bunch of times if

necessary, and more.

You will be able to use these in order to not write out as many things inside of your code as before simply because it will all be there, and the loops can get this all done for us with just a line or two of coding.

Think of it in this manner.

What if you were able to do some programming with a computer, and this helped to set it up to write out a program that would help us list out the numbers from one to thirty.

Or what if you would be able to get the code to create a really complex multiplication table.

You could go through and write out all of the lines that make this happen if you want, but is that really something that sounds like fun?

Go to that multiplication table.

That could be more than 100 lines of code by the time you go from one by one all the way to ten by ten.

No one wants to write out the codes that many times and you would get bored and have an extremely long program for something that is really simple to work with.

This is why the loops are going to be important to what we are doing.

Depending on what you are hoping to get done, and the loop type that you want to use, it is possible to get all of those lines of code done with a few simple lines of looping in the process.

If you are writing out a program and you think that you will need to have some of these loops show up, make sure to read through that section of this chapter again and learn about the different types that you can use, how they each work, and what you are able to do with them to help you add them into your programs as well.

As we can see here, there are a lot of benefits and uses that we will find when it comes to working with these loop statements and these arrays.

They are all going to be important to your codes, and there are actually a lot of times when we would be able to use this for our benefit as well in our own codes, even as a beginner.

The C++ Pointers

Another topic that we are going to spend a moment on in this chapter that kind of goes along with the arrays and the loops that we are discussing, and will be brought up a few times throughout this guidebook is the idea of the pointers.

These are easy parts of code to learn, and some programmers may see that they are a lot of fun as well.

Some of the tasks that you want to accomplish in C++ will be easier to work with when you use the pointers.

And there are even some tasks, like dynamic memory allocation, that we are not able to perform without these pointers in place.

As we already know from our work through this guidebook, every variable is basically going to be a location in the memory of our computer, and each of these locations in the memory is going to have the address defined.

We can then access this specific address with the & operator any time that we want to.

With that in mind, we can move on to some of the work that we want to do with these pointers.

The pointer is just going to be a type of variable, but it is a variable type that has its value as the address of one of the other variables.

Like any constant or variable, you need to take the time to declare the pointer before you can actually work with it.

This is pretty simple to work with, and the following code will make it possible:

Type “*var – name;

Here, the “type” part of the code is going to be the base type of the pointer.

We need to make sure that it is a valid type in C++, or this will not work.

And then the var-name part of the code is going to give us the name of our pointer.

The asterisk that is added into that is going to be used to help declare our pointer, and it is the same symbol that we are going to use when we are

working with multiplication.

When working with this statement, in particular, we will use the asterisk to help us take our variable and then designate it as a pointer for this code.

The data type that we will actually use for the pointers in question, whether we are dealing with a character, a float, an integer, or another type, is going to be the same.

This will turn into a long hexadecimal number that is going to represent a memory address to us.

The only difference that we are going to see between the pointers, along with the different types of data, is that the type of data of the variable or the constant that the pointer is going to point out here is going to be different for each part.

This is where we get into some of the fun stuff and look at how we can use our pointers.

There are going to be a few different operations that are important here, which we will do with the pointers on a regular basis.

The first step here is that we want to define the variable that we are using with the pointer.

Second, we want to make sure that we are assigning the variable address to the pointer that we want to use.

And then the third one is where we are able to gain the access that we want to the value that goes with that address.

This third one is going to sound a little bit difficult to work with, but we are able to handle it by using the unary operator of * that is going to return the values that come with our variables, and these are found at the specified address given to us by the operand we chose.

There are a few methods that we can use with this, but a coding example of how these show up is a good place to start.

The example we will send some time on now is below:

```
#include <iostream>

using namespace std;
```

```
int main () {
   int  var = 20;   // actual variable declaration.
   int  *ip;        // pointer variable
   ip = &var;       // store address of var in pointer variable
   cout << "Value of var variable: ";
   cout << var << endl;
   // print the address stored in ip pointer variable
   cout << "Address stored in ip variable: ";
   cout << ip << endl;
   // access the value at the address available in pointer
   cout << "Value of *ip variable: ";
   cout << *ip << endl;
   return 0;
}
```

There are a lot of languages that are going to work with these pointers in order to help them get things done.

But we need to spend our time focusing on how we are able to work with these pointers in the C++ language.

Pointers are going to have a lot of concepts, but luckily they are easy despite the amount of importance they will have in your code.

There are a few concepts about our pointers, though that we need to spend a bit of time on to ensure we are using them in the proper manner in our codes.

Some of the important concepts about these pointers that we still need to discuss include:

1. The Null Pointers:

The C++ language does support the null pointer.

This is going to be a constant that has a value of zero, and it is going to be defined with several standard libraries along the way.

2. Pointer arithmetic:

You will find in C++ that we can work with four operators that fit in this category of arithmetic and can be used as pointers.

These will include (++) (--), (+), and (-).

3. Pointers vs. arrays:

As you go through here, you are going to notice that there is a close relationship present when we talk about the arrays and pointers in this language.

4. An array of pointers:

It is also possible for us to go through and define our arrays so that they can hold onto the pointers that we want.

5. Pointer to pointer:

C++ will let us have some fun here by allowing for us to have a pointer on a pointer, and even more levels of this if we need it.

6. How to pass the pointers to the functions:

Being able to pass the argument is going to be important here.

And passing this argument as one of the addresses or our references will allow the argument to have some changes occur so that it can call up the function that we want along the way.

7. Return pointer from your functions:

You will find that the C++ language is going to allow us a way to use our functions to come through and return one of these pointers to our local variable, to the allocated memory that is dynamic,and to the static variables.

As we can see here, there are a lot of different ways that we are able to work with our pointers, and they can fit in well with some of the other topics that we discussed in this guidebook.

As we go through and learn more about our operators, the variables, and even the arrays and the loops that will make up our codes, we can start to see how

we can put this all together and get some good results with our codes, without it getting too complex or hard to deal with along the way either.

Chapter 6: The Operators in C++ and When to Use Them

The next topic that we are able to work with is known as the operators.

These are going to be important because though they are simple, they are going to add in a lot of power to the codes that we want to use.

There are a lot of different types of operators that we are able to focus on, and it is important to learn how each one can be added to some of the codes that we want to work with.

These operators are going to be seen as a basic or simple part of the code to work with, but they are still able to add in a ton of power to some of the codes that we want to write.

These are good to know about whether we work with the C++ language or some of the other types of coding that are out there.

With any type of programming language that you want to handle, it is important that you learn the different types of operators because they will tell the program what you would like to see get done.

They are simple symbols that we are able to handle, but they can add in more power.

While there are a ton of different operators to work with, but we are going to spend our time looking at the four most common types that work well with the C++ language.

Each one is going to be responsible for helping our code to behave in a different manner.

As we go through all of this, you will be able to figure out pretty quickly how well each one works, and what all of them are able to do.

The four types of operators that we have to spend our time on in this language include the logical operators, assignment operators, arithmetic operators, and the relationship operators.

With this in mind, we need to go through and spend a little time on each of the operator types that we are able to spend our time on.

This will help us to learn more about the operators and how they are going to

behave well inside of your own code.

The Logical Operators

To help us get started on this process, we are going to spend some of our time on the logical operators.

These logical operators are going to be the ones that you want to use in order to compare together the different parts of the code that you want to add to the system and can help you to figure out whether those parts are going to be considered true or false, whether they go together, and more based on the conditions that you are able to work with here.

Some of the logical operators that you are most likely to work with when it comes to the C++ language will include:

- (!): This operator is known as the logical NOT.

This is going to be used at any time that you would like to reverse the stated status of your operand.

So, if you have a condition that is false, you would be able to use this operator in order to make the condition true.

- (&&): This operator is known as the logical AND.

This is going to be used if you have two operands who do not equal zero.

If this happens, then the condition will end up being true.

- (||): This operator is known as the logical OR.

When you are working with this one, the condition will be true if at least one of the operands ends up being equal to zero.

The Arithmetic Operators

The second type of operator that we are going to spend some time here are the arithmetic operators.

These are often going to be seen as some of the easiest out of all the C++ operators to work with, and it is common for you to use these, along with the assignment operators, inside of your codes.

If you think back to some of the old math classes from the past, then you should be able to take a look at some of these operators and recognize what they are and how they work.

These arithmetic operators are going to use when it comes to addition, subtractions, division, and multiplication based on what you are hoping to get the program to do.

Some of the arithmetic operators that are going to happen in C++ that fit in this category and can work well in your code include some of the following

- (+): this is the addition operator that will add together two operands of your choices.
- (-): this is the subtraction operator. It is going to take the right-hand operand and subtract it from the left-hand operand.
- (*): this is the operator that makes it possible to do multiplication in the C++ language.
- (/): this operator helps you to do division in C++.
- (++): this is the increment operator. It is going to increase the value of your operand by one.
- (--): this is the decrement operator. It is going to decrease the operand value by just one.

You will find that there are a lot of options that we are able to choose from when working on these arithmetic operators.

The first thing is that we do have the chance to work with as many of these operators at a time if we would like.

So, if we have a string of ten numbers we want to add together, then this is possible to work with as well.

You can also go through and work with a few of these operators at a time, such as multiplying and dividing the parts as you go along.

You just have to work with the order of operators to make sure that you do these in the right order and get the right answers.

The Assignment Operators

Now it is time for us to move on to the third operator that works well in the C++ language.

This category is going to be the assignment operator.

This is going to be the most common out of the operators in order to make your codes work well, and it is often used to help assign the value over to the variable so that it works properly.

This assignment operator is going to be important because it is going to be responsible for helping make it easier to assign a value or a name to your variable.

It is also a good option to use when it comes to saving, searching, and so much more inside of any of the codes that you are working on.

Some of the other types of assignment operators that we are able to work with, outside of the equal sign to give a value to the variable will include some of the following:

- (=): this operator is the simple assignment operator.

It is going to assign the value of the operand on the right hand to the one that is on the left.

- (+): this one is called the Add AND operator.

It is going to add together the values from both operands and then assigns the sum of these over to the operand on the left side.

- (*=): this is the Multiply AND operator.

It is going to multiply both of the operands and then give the results over to the operand on the left side.

- (-=): this is the one that will subtract the value of your operand on the right side from the one on the left and then gives this difference to the left operand.
- (/=): this is the divide and operator.

It is going to divide the value that is on the left side from the one on

the right side and then assigns this amount to the left side.

One of the things that we need to take a look at here is how the assignment operators are going to show up in a lot of the codes that we want to use in C++.

There are some that we will use on a regular basis, like the equal sign working with our variables and assigning a value over to them.

And then there are going to be some of the options above that are not that common, and it is likely that you are not going to use them until you advance up in your coding quite a bit.

Most of the time, though, the assignment operator that you are going to work with is the equal sign.

This gives us a chance to assign the right value over the variable that we want to work with, but there can be times when we would like to add in some of the other assignment operators to help move our code along.

The Relational Operators

The next thing on the list that we need to take a look at is going to be the relational operators.

These are going to be a slightly different thing than what we have done with some of the other options, but they will ensure that we are able to check out a few of the different options that we are able to handle along the way.

Some of the options that you can work with when it comes to the relational operators include:

- (==): this is the operator that is going to check the equality of your two operands.

If they are equal, the conditions will be true.

- (>): this operator is going to check the value of your operands.

If the operand on the left side is higher than the one on the right, the condition will be true.

- (<): this operator is basically the opposite of the one above.

If you find that the value of your operand on the left side is greater than the one on the right side, the condition will be true.

- (!=): this one is going to check the equality of your two operands.

If the values are unequal, your condition is true.

- (<=): this operator is going to check whether the operand on the left side is less than or equal to the operand on the right side.

If it meets these criteria, the condition is true.

- (>=): this one is going to check if the value of the operand on the left side is greater or equal to the one on the right side.

If it is true, the condition is true.

As we can see, there are a number of C++ operators that we are able to work with.

All of them are going to work in a different manner, but you will find that these things inside of the code and can make the code more powerful to work with overall.

They may seem like a really simple addition to the code that you are working with, but they are really going to make a big difference to the codes that you are using.

Chapter 7: The Decision Control Statements

Next on the list of topics that we need to spend some time on in this guidebook are the conditional statements, which can also by the name of the decision control statements.

There will be a lot of times when you are creating one of your own programs, and the ultimate goal with this is to make sure that it can actually create some of the decisions that it needs to based on the input, for when you are not there.

This could happen when you are creating a program where the user is able to add in some of the information that they want, and then you set up the program in a manner to respond based on the various conditions that you have set up.

The reason that the program is going to behave properly here is that you are able to set up some conditions that it will base its actions on.

The decision making that happens with all of this is going to be a great way to make sure that the program is actually going to behave in the manner that you want, and it can become a more complicated kind of code.

But since this is going to be a common type of coding that we want to work with, you will gt some of the basics down quickly and find that for the type of code it is, it is easier to work with.

Of course, depending on the conditions you want to set, and what you are hoping to accomplish in your code, there are a number of decision control statements that are available to work with.

For example, we can set it up to work with a conditional statement that only allows one answer to show up if the input matches the conditions you set.

There are also conditional statements that will be able to have a list of possible answers, and it will choose the one that makes the most sense based on the input that the user provides.

Or there is even a third type of conditional statement that we are able to work with, one that is going to list out the options that the user can pick from by using a menu style and giving the corresponding result based on what the user picked from.

These are just a few of the things that we are able to do with these decision control statements.

So, with this in mind, let’s go through and take a closer look ad the options you have when working on these conditional statements.

How to Work with the Switch Statements

So, the first of these conditional statements that we can spend our time focusing on will be known as the switch statements.

These are good conditional statements to work with because they can help us to check out whether there is some equality in our variables, and this is going to be checked against the cases or values that you set up in your code.

The variable that you would be checking here is then going to be compared against a variety of cases to see what is correct or not.

This is easier to work with than it may seem in the beginning, so let's dive in and look at the syntax that we are able to use with the switch statement below:

```
Switch(expression){
        case constant-expression:
        statement(s);
        break; //optional
        case constant-expression:
        statement(s);
        break; //optional
//you can add in as many of these case statements as you would like
Default: //Optional
statement(s);
}
```

When we bring out these switch statements, there are going to be a few rules that we need to follow in order to make sure that these work well in our codes.

First, the expression that we are going to use with the switch statement is going to be either an integral or an enumerated class type.

In addition to all of this, it is also able to belong with a class that has a function for conversion.

The good news that we are going to see here is that when we work with the C++ language, there will not be any limits that show up when it comes to the number of case statements that you can add into the syntax above, so you can choose to make the switch statements as long or as short as you would like along the way based on your code.

The thing that we need to remember here is that at the end of the right parts, a colon has to be added to all of the values to make sure the code works well.

After the variable has some time to find a value that is going to be equal to it, then it is time for that variable to keep running until it is able to find the break statement.

When this break statement is found, then the switch is going to be able to stop what it is doing.

The control flow from this part is going to be able to pass onto the next part of our code as well.

You do not have to go through here and add in a break statement to the cases if it is not needed.

If you miss this or decide not to add it in, then the control flow is simply going to pass over to the following part.

Using the If Statement

Now that we know a little bit more about how to work with the switch conditional statements, it is time for us to move on to looking at the second kind of conditional statement, which is known as the if statement.

One of the most basic things that you can do with these statements as a whole is this one.

They are going to be based on the simple idea of true or false.

If the system decides that the input it gets is going to match up to the conditions that you are able to set, then the condition is true, and the program will be able to run the part that you have set up here.

We need to look at an example of how this is going to work.

Let's say that you set up a system that is going to ask the user what 2 + 2 is.

The user then puts in the right answer of 4 here.

This would be seen by the compiler as true, and then the message would be able to pick out a statement based on what you add there, like "That is Correct!" and then the message is going to show up on your screen.

Any time that you go through the process of having the user add in the necessary input, there is some risk, though.

If they work through this and add in an input that meets your conditions, then the code will proceed with the statement, or whatever else you add to that part of the code.

But there are also times when the user could add in the wrong answer.

Going back to our previous example of asking the user what 2 + 2 is, if they answer with 5, then the compiler will see that this is not the right answer, and the number is not going to be able to match up to the conditions you set, resulting in a false.

Since we are working with the if statements here and they are seen as the most basic out of all the conditional statements that are out there, you may find that this is not going to have a method to help answer for the false input here.

If the user gets the problem above and puts in any answer that is not 4, then the if statement is going to see this as being false, and then the screen is going

to be left blank since nothing can happen.

This is going to turn into a problem for most of the codes that you want to write out, so we can imagine that it is not often that you will work with the if statement.

The If Else Statements

Next on our list is going to be the if else statements.

While there are some times when we want to work with the if statements, we can already see that there are a few limitations that come with it as well.

You are going to run into some issues with this any time that the user ends up adding in the wrong answer to the program, and then the program sees this as a false statement based on the conditions that you added into it.

The good news is that there is a way to fix this issue, and it is where we are going to introduce the if else conditional statement.

A programmer is going to find that the if else statement is able to work in a lot of different situations.

Not only does it allow for the user to put in an answer that may not meet your conditions, but there are also times when you may want it to let the user pick from a few different options that we can start out with.

For example, the if else statement is going to help you to do this, and you will find that the syntax is going to be simple.

The syntax that you want to use to create your own if the else conditional statement is going to include the following:

```
if(boolean_expresion)
{
        //statement(s) will execute if the boolean expression is true
}
Else
{
        // statement(s) will execute if the boolean expression is false
```

You will be able to add in as many of these points into the if else statement as you would like.

If you would like to work with your own program that is able to set apart people, so they are in five or six different age groups, you would be able to

do this with the help of these if else statements.

You would just need to add in some more of the “else” keywords to make this happen.

We are able to take a look at how we are able to make this work while trying to keep it as simple as possible.

Let’s say that you are working with a program that has 2 + 2 as the part that shows up on the screen, and then you want the user to come up with the answer.

If the person guesses that the answer is 4 and puts that in, you will set up the if else statement so that the first part is going to be the true statement, and then you can add in the message that you would like to work with.

However, if your user decides that the answer to this is anything other tana the number four, the program is set p to see this as a false answer thanks to some of the conditions that you already set.

Instead of going to a blank screen like before, we are going to be able to change this up so that there is a second option that is found in our code, and we can use this to get another statement to show up. For example, we could get it set up to say something like “sorry, that is incorrect” to come up on the screen.

As we can see already, using this conditional statement is going to help us get a lot more freedom when it comes to the things that we can do with the program.

You are able to keep it as simple as the example that we did above, or you can add in some more to it to add in some complexities and more and will get it to work with some of the codes that you worked with as well.

One thing that we should keep in mind as we work with these if else statements is that it is possible for us to go through and add in some more if else parts to this, and it is even possible to go through and add inside a few of the if else statements to each other if you would like.

If you are a beginner, this can be more complex to work with, but with a little bit of practice and learning how to use the syntax that is above.

You will find that it is going to add in some good power to the process and will make it so much easier to work with the different tasks that you are

hoping your code is able to accomplish.

Working with the if statements and also with some of the if else statements will make your experience with coding that much better.

It is going to allow the system to make some good decisions based on what the user is able to put into the system, without the programmer going through and having to guess all of the possible outcomes ahead of time.

To help you work with these conditional statements and how they are going to work, take some time to look through the different options of these statements above so that you are able to utilize them for your own needs as well.

How the Elif Statements are Different

The fourth type of conditional statement that we need to take a look at here is going to be the elif statements.

These are going to be kind of the same idea that we see with the if else statements and more, but they can take this idea a little bit further along the way to add in some of the strength that your coding needs.

These statements are going to actually list out the different options that you are looking at for your user to focus on.

Think about it kind of like a menu in a game with all of the options and choices that the user is able to pick from.

You can make your menu have a few different things on it, and then the user is free to make any of the choices that they want.

The elif statement is going to be a really good option to work with, and it adds in a ton of functionality to the codes that you want to work with.

The user to your program will be able to help us out to pick from all of the options that are there.

They can't make any decisions that are not on that menu though with this one.

And in some cases, we are able to add in a catch-all that the user is able to pick from if they are not that interested in some of the other results that your dd into the menu.

This is a pretty simple option to work with, and you will have the option to go through and add in as many of these elif statements as you would like.

Sometimes the code that you work with will just need a few options to make it work, and other times there are a lot of options, and you can make it as long as you would like.

It is also possible to go through this and add in some statements so that the user is able to pick out each option that they want.

The statement that you are going to place here is going to vary based on what kinds of conditional statements you would like to use here.

For example, let's say that we are working with a game or another kind of program, and your goal is to get the user to go and pick out the specific type

of pizza that they would like to enjoy with the program.

You can go through and list out as many of the different types of pizzas as you would like, but we are going to keep this as simple as possible and just go with four choices, namely cheese, veggie, sausage, and pepperoni for some of our needs.

The user is able to then pick from those pizzas, and then the statement that you set up to go with each one is able to execute once this choice has been made.

It is even possible to make this elif statement work where you can have the user not pick out any pizza topping or any pizza at all, and they can choose to just have a drink at this part of the program.

As we can imagine here, there are a lot of times when we will want to work with this elif statement, and it can be a lot of fun to add into some of the programs that we want to create.

And you have the flexibility with this kind of coding in order to add in just a few options, or you can add in a ton of options based on how your specific code is supposed to work.

And there you have it!

This chapter took a look at some of the conditional statements that you are able to spend your time on, and how you can make some of these work for your own needs.

There are a ton of conditional statements out there, and there are a lot of times when we are able to use these in some of our own codes as well.

All four of the conditional statements that we had a chance to talk about in this guidebook are going to be important, and there are a ton of different things that we are able to do with each one.

We can find that we are able to do a lot of things when it comes to using our conditional statements, and the one that you will pick out is going to depend on what your goals are with the program, and what you are looking to do with it in the process.

Of course, there are not a lot of times when we are going to want to use them if statements, simply because they can leave us with the potential of a program that is not going to work all that well.

No one wants to have a program that is not going to showcase off anything, even when the answer is wrong, and the blank screen that could show up is never going to look all that professional.

But it is still an important option to learn how to use because it allows us a chance to go through and really work with some of these statements and get more familiar with it in the process.

The other three conditional statements can show up a lot in the codes that we want to work with as well.

These are going to be important to ensuring that our codes are able to make some of their own decisions, even if we are not sure what the results or the input are going to be from the user.

These can take over, and we can set up the conditions that will tell them how to behave along the way.

This is going to make it easier for us to ensure that the codes we have are always going to behave well.

There are a lot of different ways that we can bring in the conditional statements and get them to work the way that we want.

There is a lot of power that comes with these statements, and they will allow us to get the program to work the way that you would like.

You can get it to make some good decisions for you and get it to behave in the manner that you would like as well.

Before you move on to some of the other topics that are out there concerning the C++ language, take some time to practice these so that you can bring them out in any codes that you would like.

Chapter 8: Working With Inheritances

The next thing that we are going to take a look at here is going to be the inheritances.

These are a bit more complex than some of the other stuff in this guidebook, but at this point, you are ready to take it on and really work with something that is more advanced.

You will find that wen these inheritances come out, we are going to see some of the beauty that is there with the OOP languages that we talked about before, and we can actually go through and reuse parts of the code that we want to work with.

One of the most important out of all the concepts that come with OOP languages is that of the inheritance.

This idea is going to make it easier for us to define a class based on the terms of another class in our code.

This is useful because it allows us to have an easier way to create and maintain one of the applications that we want to work with.

It is also going to be useful in providing an opportunity for us to reuse the functionality of the code and to get the implementation done faster than ever before.

Any time that we want to go through and create a class, rather than having to go through and write out a brand new members of data and members of functions each time that you do it, the programmer is able to designate that the new class is supposed to inherit the members of a class that is already found in the code.

This can make it a lot easier than having to redefine the members and the functions as you go through the code again and again.

The class that we are basing our work on or the existing class is going to be known as the base class.

Then the new class that we are trying to create, and the one that will take that information and the data or the functions out of the base class, is going to be known as the derived class.

The idea that we are going to see with the inheritance is that it implements the “is a” relationship.

For example, it can work with the idea of a mammal is an animal; the dog is a mammal; hence dog is an animal as well.

This is going to kind of simplify the process that we are working with here, but still, give us a good idea of what we are working with along the way.

The Base and the Derived Class

With this in mind, it is time for us to go through some of the particulars that we are able to do when it comes to working with the classes that are at hand.

Remember that we are going to have two classes in mind when we do this, the base class and the derived classes.

These are going to work together to help us write out the different codes that we want to see work well in all of this.

A class can easily be derived from more than one class if we would like to use it in this manner.

When this happens, this means that it is able to inherit data and functions from more than one base class as we go along.

To help us define a derived class, we are going to work with a class derivation list to help specify the base class or base classes that we want to work with as we go through this process.

We can make this as simple or as hard as we would like based on some of the work that we are trying to accomplish.

For example, we are able to work with defining a derived class, and this can be done by using the list of class derivation to help specify the classes to help us out.

The class derivation list names one or more base classes that we want to work with.

And the form that helps us to do this is below:

Class derived-class: access-specifier base-class

There are a few things that we need to keep in mind when we are working with this one.

Where you see the access specifier part of the code, we need to go through and choose whether it is public, protected, or private.

All of these are going to have some important meanings in the code, and you have to decide which one you would like this class to have.

When we are working with the idea of public access, it means that we are going to work with a class that is open for anyone to use.

Other classes in our code are able to come in and access that one or use it, and there are not that many limitations on what can happen with the class.

This is usually not the option that we want to go with because it leaves that class vulnerable to issues and even attacks, but in some cases, it is going to be the right one for us to choose to work with.

Then we have protected access.

This one is kind of the same in that it is open to quite a few things.

But we are going to put it under some protections and limitations so that it is not openly available to everything and everyone.

This one will ensure that only the parts of the code that we want to have access to this class will actually be able to access it at all.

And then we are able to end this one with the private option.

This one is going to be the most restrictive, and you should only use it when you know the exact class or part of the code that should access it, and you want to make sure that no other part is able to do so.

Along with our access-specifier part of the code, you will notice that the base-class is simply going to be the name of the class that we defined earlier.

If you find that the access-specifier is left blank or you forget to add something there, keep in mind that the default is going to be private in order to keep things safe.

If you definitely want to change this up and you do not want this to be private, then this is something that you have to change.

Now it is time for us to go through and look at an example of how we are able to work with this and get it to work for our needs.

We are going to look at inheritance, and the base class that we want to focus on will be Shape, and then the derived class is going to be Rectangle.

The code that we are able to use to make this one happen is below:

```
#include <iostream>

using namespace std;

// Base class
```

```
class Shape {
   public:
      void setWidth(int w) {
         width = w;
      }
      void setHeight(int h) {
         height = h;
      }
   protected:
      int width;
      int height;
};
// Derived class
class Rectangle: public Shape {
   public:
      int getArea() {
         return (width * height);
      }
};
int main(void) {
   Rectangle Rect;
   Rect.setWidth(5);
   Rect.setHeight(7);
   // Print the area of the object.
   cout << "Total area: " << Rect.getArea() << endl;
   return 0;
}
```

Take some time to type this code into your compiler and see what is going to happen.

You should find out what the total area of it all is going to be, and you can work from there.

As we can see here, while we did work with an example of an inheritance, it was not that bad to handle, and it is a great option that will ensure that we are able to reuse parts of the code, get the implementation of the code to be easier, and will ensure that we actually get a chance to work with the code without wasting a lot of time in the process.

The Access Control of Our Inheritance

The next thing that we need to take a look at is access control and inheritance.

A derived class is able to access all of the parts of the base class that are not private at the time.

This means that the base class means that the members of the base class that should not be accessible to all of the member functions of the derived classes are going to be declared as private for the base class when we are doing this as well.

You will notice as we go through this that the derived class is going to be able to inherit all of the base class methods.

There are going to be a few different exceptions that we are able to use with this one based on what results we are hoping to get. Some of the exceptions that we need to remember include:

1. The friend functions that are found with our base class.
2. The overloaded operators that are there with our base class.
3. The destructors, the constructors, and any of the copy constructors that are going to happen with our base class.

We need to make sure that we pay special attention to some of the different parts that show up with our access control.

If there are some restrictions that are found in our base class, then this is going to step in and cause some issues with the inheritance that we are working with as well.

And the child or the derived class is going to notice this along the way.

Type of Inheritances

While we are here, we need to take a closer look at the types of inheritances that we are able to work with.

When we try to derive a class from one of our base classes in the first place, then the base class has the possibility of being inherited through a private, a protected, or a public inheritance.

The type of inheritance that you use is going to be specified through the access-specifier, as we talked about above.

Now, you will find that it is not common for us to work with an inheritance that is private or protected, though it is possible to do this.

Just be aware that there are going to be some issues along the way, and these are not as easy to work with.

The method that we are most likely to work with here will be a public inheritance.

Even though the public option is going to be the most common for us to work with, we need to take a look at some of the rules that are going to happen when we work with the different types of inheritances.

The different rules that we need to remember when we are working with this one include:

1. The public inheritance:

When we want to derive a new class from a public base class, the public members that are found in our base, or original, class, are going to become public embers of the derived class.

And then the protected members that are found in our base class will also become protected members of some of the derived classes that we make out of it as well.

a. Was we can guess here, we are going to find that the private members that are found with our base class are never going to be accessible directly from the derived class.

We can access them through here.

We just need to do some calls to the protected and the public

members of our base class to make this happen.

2. The protected inheritance.

The second option that we are able to work with is protected inheritance.

When we want to derive a new class from a base class that is protected, the protected and the public members of the base class are going to end up as protected members of the derived class that we want to work with.

3. Private inheritance:

The final type of inheritance that we are going to work with is private inheritance.

When we are trying to derive a new class from a private base class, both of the protected and the public members of our base class are going to go over to the derived class, but then these are going to be turned into some of the private members instead.

As we can see, all of these are going to be a bit different from one another.

And this is why we need to be careful about the kind of inheritances that we are going to work with.

If you are in a different type of inheritance, then the members are not always going to work the way that you want.

Double-check what you are in, and make sure that this is set to the right kind so that it actually works the way that you want.

Doing a Multiple Inheritance

The last thing that we want to take a look at here when it comes to some of the inheritances that we are able to do with the C++ language is the idea of handling multiple inheritances.

It is possible for one of the classes in this language to go through and inherit members that come from more than one class.

The extended syntax that we are able to use with this one is going to be below:

Class derived-class; access baseA, access baseB, …

We are able to go through this and add in as many of the base classes as we want, but make sure only to use the necessary ones, or you are going to end up with a big mess in the process.

The same rules with the access that we have talked about before are going to matter with this part of the process as well.

Where the access is one of private, protected, or public and would be given for every base class, and then they will also be separated out using a comma, with the same syntax that we were talking about above.

A good way to see how these multiple inheritances are going to work is to do a bit of practice with them as well. take a look at the code below and see how we are able to create one of these, or at least create a new derived class based on a few different base classes as well:

```
#include <iostream>
using namespace std;
// Base class Shape
class Shape {
   public:
      void setWidth(int w) {
         width = w;
      }
      void setHeight(int h) {
```

```
        height = h;
    }
  protected:
    int width;
    int height;
};
// Base class PaintCost
class PaintCost {
  public:
    int getCost(int area) {
        return area * 70;
    }
};
// Derived class
class Rectangle: public Shape, public PaintCost {
  public:
    int getArea() {
        return (width * height);
    }
};
int main(void) {
   Rectangle Rect;
   int area;
   Rect.setWidth(5);
   Rect.setHeight(7);
   area = Rect.getArea();
   // Print the area of the object.
   cout << "Total area: " << Rect.getArea() << endl;
```

```
    // Print the total cost of painting
    cout << "Total paint cost: $" << Rect.getCost(area) << endl;
    return 0;
}
```

Take some time here to type in the code and see what results you are able to get in the process.

The answer should tell us the total area, while also giving us the total paint cost that we should deal with.

This is just one of the examples of what we are able to do with multiple inheritances, but as you can see, it is a lot of fun to work with and is pretty easy as well.

There are numerous times when you will want to go through and work with an inheritance in some of your own codes.

It is a simple process to work with, and when it is done right, it is going to be really helpful for ensuring that we get the code set upright and even saving a lot of time in the process.

Keep in mind that these inheritances are going to be unique because they only work in the OOP languages that we talked about.

If we are not working with one of these types of languages, then the basic ideas that come with this kind of language will not really work, and we are going to end up with kind of a mess if we try.

But since the C++ language is going to be seen as one of these OOP languages, it makes sense that one of the benefits that we are able to utilize with this one will allow us to work with inheritances.

And while it is going to sound hard to take one class and use that information in order to create a new base class of our choosing, the codes that we did above will show us how this works and ow simply this can be to work with.

There are a lot of benefits to working with the inheritances that we discussed in this chapter.

They can help to clean up the code that we want to work with; they will ensure that we are able to reuse parts which will save a lot of time in the

process, and they will really speed up how well your code is going to be able to implement as well.

If you need to reuse some of your code or you are looking to add in some more features that are found in an older part of the code to the new class that you are creating, then the inheritances are the way to get this done.

Chapter 9: The Idea of Polymorphism

Another thing that we need to spend some time taking a look at is the idea of polymorphism.

This is a unique idea that is a bit different than what we are able to see with some of the other options out there, but it is going to really help us to get some good results on certain parts of our code along the way.

With this in mind, let's take a look at how to work with polymorphism and how it is going to fit in with some of the codings that you want to work with.

To start, the word of polymorphism means that it will have many forms.

For the most part, we are going to see that polymorphism is going to occur when there is already a hierarchy of our classes in the code, and when all of these classes are going to be related, thanks to the process of inheritance that we talked about above.

C++ polymorphism means that a call to a member function is going to cause a different function to actually be executed.

The function that is going to be executed here is going to depend on the object type that would invoke the function to start with.

A good way to see how all of this polymorphism is supposed to work in our codes, we can take a look at an example.

This example below is going to show us where a base class has been derived by two other classes as well:

```
#include <iostream>

using namespace std;

class Shape {

  protected:

      int width, height;

  public:

    Shape( int a = 0, int b = 0){
```

```
        width = a;
        height = b;
     }
     int area() {
        cout << "Parent class area :" <<endl;
        return 0;
     }
};
class Rectangle: public Shape {
  public:
       Rectangle( int a = 0, int b = 0):Shape(a, b) { }
       int area () {
        cout << "Rectangle class area :" <<endl;
           return (width * height);
     }
};
class Triangle: public Shape {
  public:
     Triangle( int a = 0, int b = 0):Shape(a, b) { }

       int area () {
        cout << "Triangle class area :" <<endl;
           return (width * height / 2);
```

```
    }
};
// Main function for the program
int main() {
   Shape *shape;
   Rectangle rec(10,7);
   Triangle  tri(10,5);
   // store the address of Rectangle
    shape = &rec;
   // call rectangle area.
   shape->area();
   // store the address of Triangle
   shape = &tri;
   // call triangle area.
   shape->area();


   return 0;
```

When you have finished with compiling the code above, and you give it some time to execute, you should get the result of the parent class area there.

But both parts are going to be left blank.

The reason for this incorrect output is that the call of the function area is going to be set once by the compiler as the version that is defined in our base class.

We are able to call this the static resolution of the function call, or the static linkage.

What this means for us is that the function call is going to be fixed before we even go through and execute the program we want to work with.

This is also going to be something that we can call early binding because the area() function is set during the compilation of the program. But now we need to move in a different direction and make some small modifications to our program. In this one, we are going to precede the declaration of the area() in our Shae class with the virtual keyword.

The way that this code is going to look is below:

```
class Shape {
  protected:
    int width, height;
  public:
    Shape( int a = 0, int b = 0) {
      width = a;
      height = b;
    }
    virtual int area() {
      cout << "Parent class area :" <<endl;
      return 0;
    }
};
```

After we are able to make this small modification, and then we are able to go through and compile and then execute the code again, it is going to give us a slightly different result in the process.

We are going to get the Rectangle class area and the Triangle class area instead.

With this one, the compiler is going to take a look at the contents that are in the pointer instead of its type.

Because the addresses of our objects of the rec and the tri are going to be

stored in the *shape, then the respective area of our function is going to be called up here.

As you can see with this one, each of the child classes that we work with is going to have a separate implementation of our function area.

This is usually the way that we are able to work with polymorphism.

You can also come with different classes with a function of the same name, and even with the same parameters, but it is going to need to work with a different implementation to make this work.

Another thing that we are able to work with here is going to be the virtual function.

This is going to be a function that is found in a base class that can be declared with the “virtual” keyword, as we talked about above.

Defining the base class as one of these virtual functions, with the help of one of the other versions in your derived class, is going to signal to the compiler that we do not want it to work with the static linkage for this particular function at all.

What we do want to do with this one, though, is to have the selection of the function to be called up at any given point in our program, and we want it to be based on the object type that is being called.

This sort of operation is going to be known as late binding or dynamic linkage.

In addition to all of this, it is possible that you will want to go through and include your own virtual function in the base class that you work with so that it can be redefined in a derived class to help suit the objects to really suit all of the objects that are in that class.

But to make sure this works, we want to make sure that there are no meaningful definitions that you could provide to the function in the base class.

This means that we are able to take our virtual class and change it over to the base class if that works the best for our needs.

Conclusion

Thank you for making it through to the end of ***C++ for Beginners***, let's hope it was informative and able to provide you with all of the tools you need to achieve your goals whatever they may be.

The next step is to start writing out some of the different programs and codes that you want with the C++ language.

Even if you are just starting and you just want to spend your time practicing with some of the codes that are in this guidebook, you are still on the right path to seeing some great results with this language.

The more practice you are able to get along the way with your own coding, the easier it will become to write and create your own codes along the way.

The C++ language is a unique option to work with, one that will help us to create some really powerful and strong codes, and the functionality that is found with this language is unlike any other that we are able to work with.

And inside this guidebook, we took our time to explore this in more detail so that we could really see how things are meant to work, and what we will be able to get out of the process of using this as well.

We started out this guidebook with a nice introduction to how we are able to work with the C++ language, and why it is one of the best options to choose from.

And then, we took you to step by step through the process of writing out some codes and even creating a program.

This is there to show you that, even as a beginner who has never had a chance to work with coding before, that even in the first few chapters of a coding guidebook, you can get right in and create some of your own programs.

Once you have all of that down and some confidence in your own code writing progress, it is time to go into some more of the specific codes that we can write, and this guidebook delivered.

We spent time looking at how to work with the variables, the loops and arrays, the operators, the decision control statements, the idea of inheritances in your code, and even with polymorphism.

All of these can be important in some of the codes that you want to write along the way, and when we are done with this guidebook, you should feel ready to handle these and add them to your own codes as well.

As we can see, there is so much that we are able to do when it comes to working with the C++ language.

It is one of the best coding languages out there, whether you are a beginner or you are someone who has worked with programming and coding for a long time.

When you are ready to start your own coding journey, and you want to be able to create some of your own programs in no time.

Make sure to check out this guidebook and get started with the C++ language.

Finally, if you found this book useful in any way, a review on Amazon is always appreciated!

All Books published by Julian James McKinnon:

Linux for Beginners: A step-by-step guide to learn architecture, installation, configuration, basic functions, command line and all the essentials of Linux, including manipulating and editing files
Hacking with Kali Linux: A Step by Step Guide with Tips and Tricks to Help You Become an Expert Hacker, to Create Your Key Logger, to Create a Man in the Middle Attack and Map Out Your Own Attacks
Hacking for Beginners: A Step by Step Guide to Learn How to Hack Websites, Smartphones, Wireless Networks, Work with Social Engineering, Complete a Penetration Test, and Keep Your Computer Safe
C# For Beginners: A Step-by-Step Guide to Learn C#, Microsoft's Popular Programming Language
C++ for Beginners: A Step-by-Step Guide to Learn, in an Easy Way, the Fundamentals of C++ Programming Language with Practical Examples
SQL For Beginners: A Step-by-Step Guide to Learn SQL (Structured Query Language) from Installation to Database Management and Database Administration
Python Programming for Beginners: A Step-by-Step Guide to Learn one of the Most Popular and Easy Programming Languages. Learn Basic Python

Coding Fast with Examples and Tips
Data Science with Python: The Ultimate Step-by-Step Guide for Beginners to Learn Python for Data Science
Arduino: Learn how to Create Interactive Electronic Objects, Setting up your Board, Discover how Coding Works, Create your Circuit plus all the essentials of Arduino Programming (For Beginners)
Raspberry Pi: A Step-by-Step Guide for Beginners to Learn all the essentials of Raspberry Pi and create simple Hardware Projects like an Arcade Box or turning your Device Into a Phone