techgo

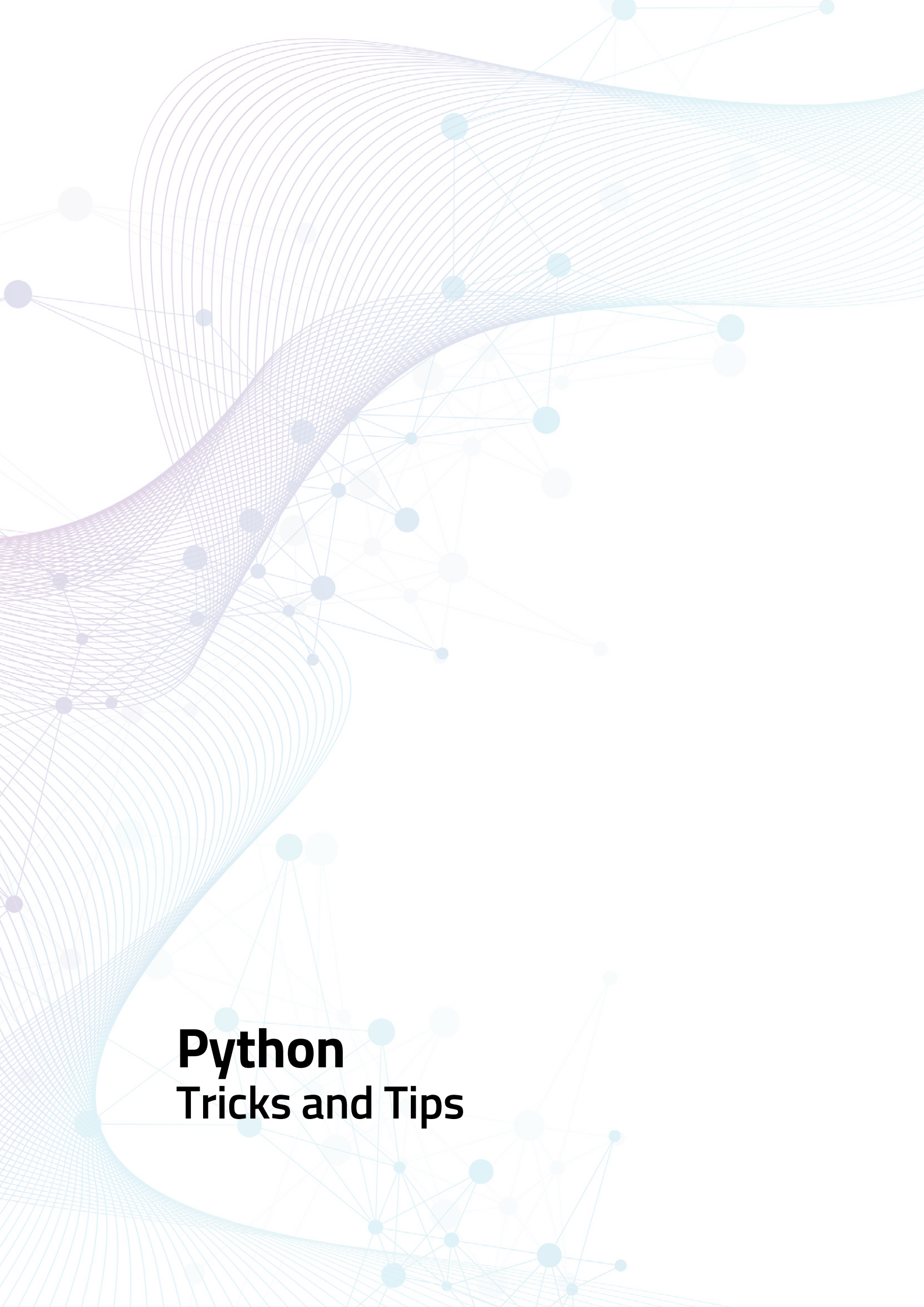
Python Tricks and Tips

Gain Insider Skills

Next level Secrets & Fixes

Advanced Guides & Tips

Rediscover Your Device



Python
Tricks and Tips

Python Tricks and Tips
ISBN: 978-1-912847-52-5

Foreword

Welcome back...

Having completed our exclusive For Beginners digital guidebook, we have taught you all you need to master the basics of your new device, software or hobby. Yet that's just the start!

Advancing your skill set is the goal of all users of consumer technology and our team of long term industry experts will help you achieve exactly that. Over this extensive series of titles we will be looking in greater depth at how you make the absolute most from the latest consumer electronics, software, hobbies and trends!

We will guide you step-by-step through using all the advanced aspects of the technology that you may have been previously apprehensive at attempting. Let our expert guide help you build your understanding of technology and gain the skills to take you from a confident user to an experienced expert.

Over the page our journey continues and we will be with you at every stage to advise, inform and ultimately inspire you to go further.

About the Publisher

From its humble beginnings in 2004, the BDM brand has quickly grown from a single publication produced by a team of just two to one of the biggest names in global tech print and digital publishing, for one simple reason. Our passion and commitment to deliver the very best product to the marketplace.

While the company has grown with a portfolio of over 500 publications crafted by our international staff of respected industry veterans, the foundation that it has been built upon remains the same. That being to create the best quality, fully independent, user friendly and, most essentially, 100% up-to-date content possible.

Delivering not only market leading publications but also piece of mind to our readers, so that they have the very best foundation to build their knowledge, confidence and understanding of their new software and hardware. Our regular readers trust BDM, as should you.

How to use this book

This book has been designed for you to progress through the coreconcepts and fundamentals of use, through to more advanced elements, projects, and ideas. There's something for every style of reader, and for every type of user; there's probably even a few terrible jokes dotted within the pages. So here's how to get the best from it.

Step 1

Don't skip - While it's fun to see what's coming up later in the book, it does make understanding what you're reading more difficult. After all, you wouldn't start reading a book on speaking French, then skip further in without first learning proper grammar, sentence structure and so on. The same applies here. Take your first read-through page by-page, once you've mastered the book, then you can return to key concepts whenever you need.

Step 2

Ever-Changing - While every effort has been made to ensure that this book is up to date, there's no knowing what updates may occur over time. While some companies do offer an accurate roadmap of their future development of a product, it's not always written in stone. For example, an app available for Windows 10 now may not be available with the next update of the operating system. It's up to Microsoft to decide whether they want to drop it for one reason or another. The same, to some extent, applies here. However, we continually update the content in this title, so it's as accurate as possible.

Step 3

Follow the Steps - An obvious one, this. Following the steps from one onwards, in most tutorials in this book, will ensure that you get the result that's intended. If you skip steps, then you may miss out on something important, and not understand how it works later in the book. The temptation to skip something you already know is often too much, but stick with the logical progression of the steps and you'll get the most from what's on offer.

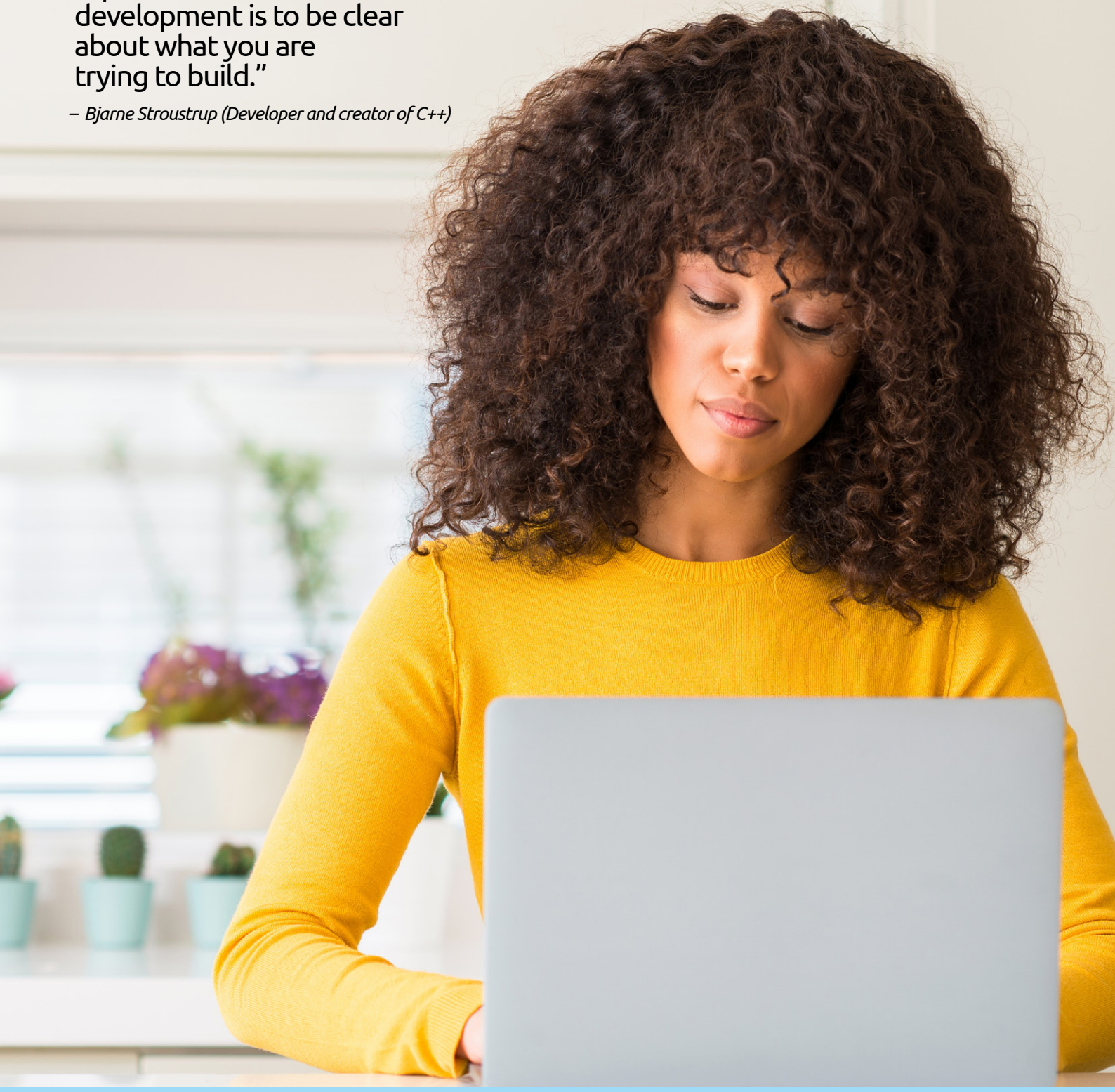
Step 4

Have Fun - Learning a new skill is supposed to be fun. We had fun writing the book, and hopefully you'll have fun reading it and applying new skills. Everyone learns at a different pace, so take your time, digest the tutorials, and keep returning to key concepts if you feel the need to master any element within these pages. The content in this book isn't something we're going to be testing you on, so have fun and enjoy the art of learning something new. And if you create something amazing after reading this book, then let us know.

Contents

"The most important single aspect of software development is to be clear about what you are trying to build."

– *Bjarne Stroustrup (Developer and creator of C++)*



"The most important property of a program is whether it accomplishes the intention of its user."

– C.A.R. Hoare
(Computer scientist, developer of Quicksort)

Say Hello to C++

C++ is an excellent, high-level programming language that's used in a multitude of technologies. Everything from your favourite mobile app, console and PC game to entire operating systems are developed with C++ as the core, together with a collection of software development kits and custom libraries.

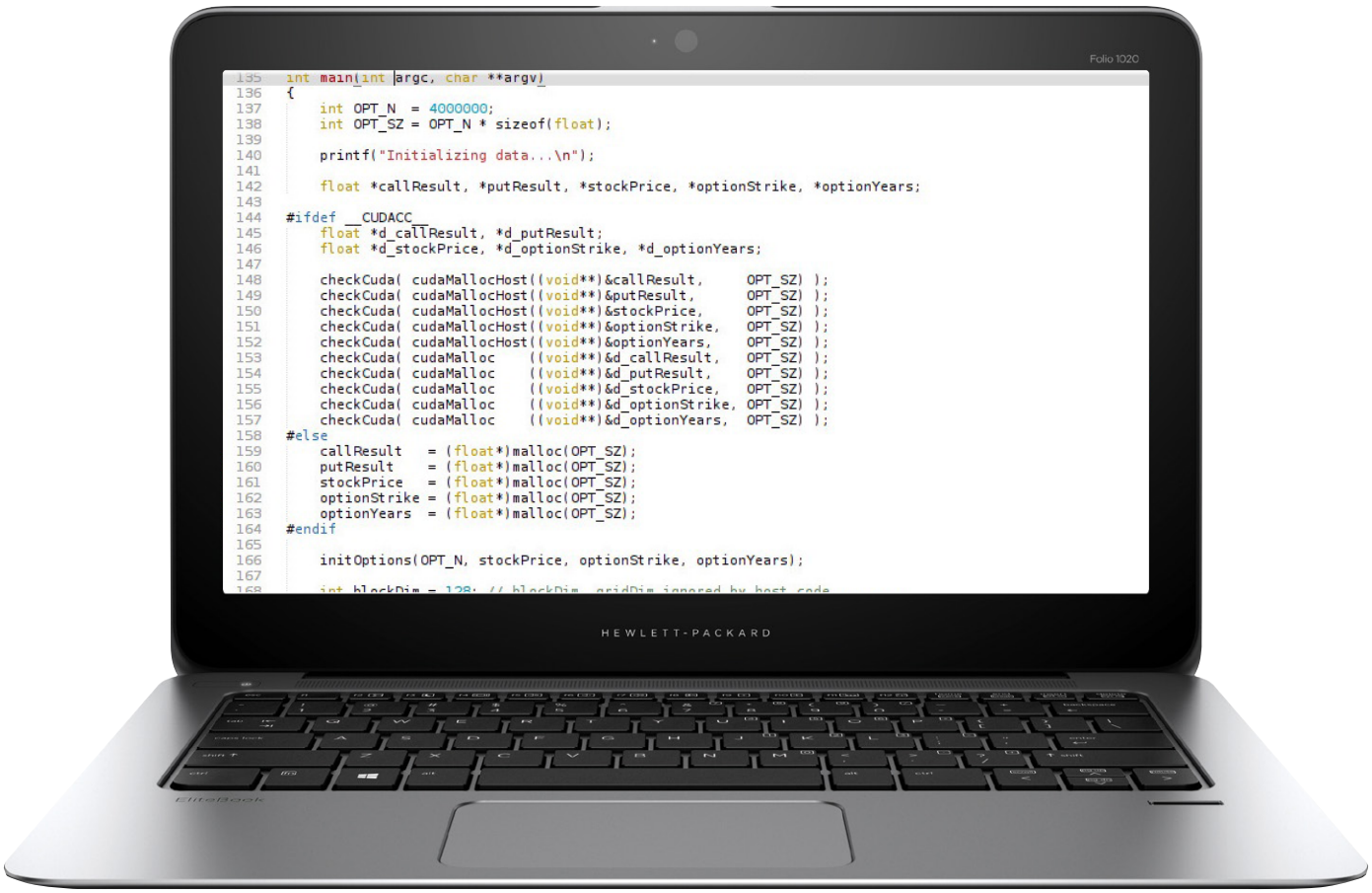
C++ is the driving force behind most of what you use on a daily basis, which makes it a complex and extraordinarily powerful language to get to grips with. In this section, we look at how to install a C++ IDE and compiler on your computer.

Why C++?

C++ is one of the most popular programming languages available today. Originally called C with Classes, the language was renamed C++ in 1983. It's an extension of the original C language and is a general purpose object-oriented (OOP) environment.

C EVERYTHING

Due to how complex the language can be, and its power and performance, C++ is often used to develop games, programs, device drivers and even entire operating systems.

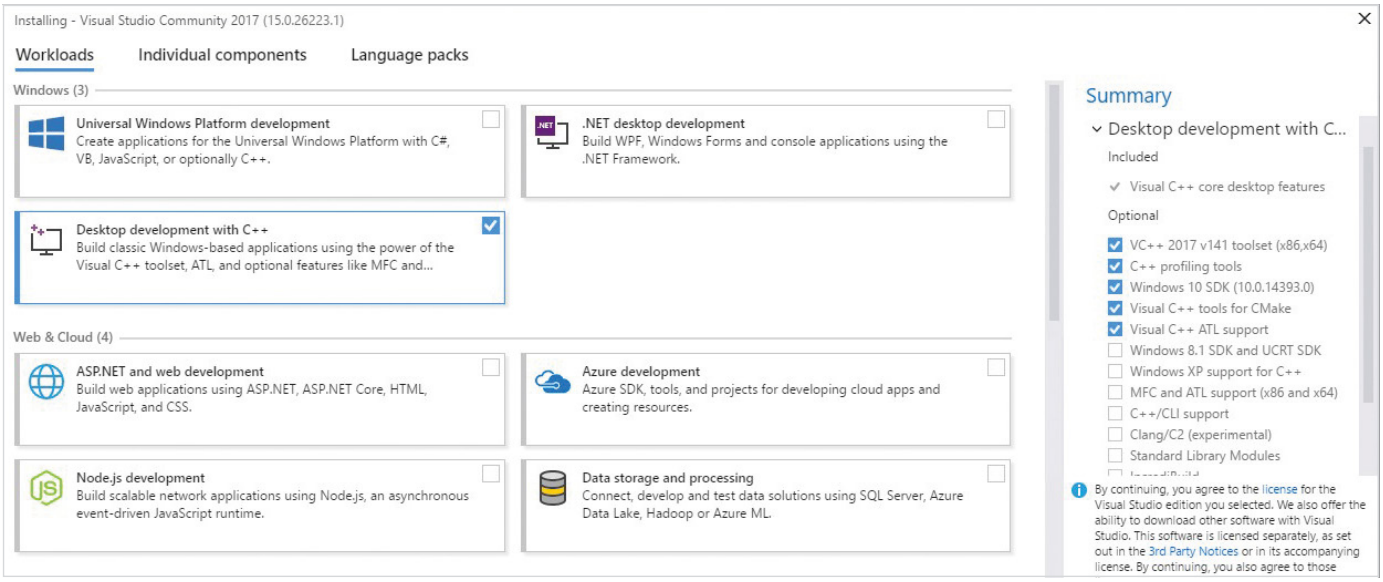


C++ code is much faster than that of Python.

Dating back to 1979, the start of the golden era of home computing, C++, or rather C with Classes, was the brainchild of Danish computer scientist Bjarne Stroustrup while working on his PhD thesis. Stroustrup's plan was to further the original C language, which was widely used since the early seventies.

C++ proved to be popular among the developers of the '80s, since it was a much easier environment to get to grips with and more importantly, it was 99% compatible with the original C language. This meant that it could be used beyond the mainstream computing labs and by regular people who didn't have access to the mainframes and large computing data centres.

C++'s impact in the digital world is immense. Many of the programs, applications, games and even operating systems are coded using C++. For example, all of Adobe's major applications, such as Photoshop, InDesign and so on, are developed in C++. You will find that the browser you surf the Internet with is written in C++, as well as Windows 10, Microsoft Office and the backbone to Google's search engine. Apple's macOS is written largely in C++ (with some



Microsoft's Visual Studio is a great, free environment to learn C++ in.

other languages mixed in depending on the function) and the likes of NASA, SpaceX and even CERN use C++ for various applications, programs, controls and umpteen other computing tasks.

C++ is also extremely efficient and performs well across the board as well as being an easier addition to the core C language. This higher level of performance over other languages, such as Python, BASIC and such, makes it an ideal development environment for modern computing, hence the aforementioned companies using it so widely.

While Python is a great programming language to learn, C++ puts the developer in a much wider world of coding. By mastering C++, you can find yourself developing code for the likes of Microsoft, Apple and so on. Generally, C++ developers enjoy a higher salary than programmers of some other languages and due to its versatility, the C++ programmer can move between jobs and companies without the need to relearn anything specific. However, Python is an easier language to begin with. If you're completely new to programming then we would recommend you begin with Python and spend some time getting to grips with programming structure and the many ways and means in which you find a solution to a problem through programming. Once you can happily power up your computer and whip out a Python program with one hand tied behind your back, then move on to C++. Of course, there's nothing stopping you from jumping straight into C++; if you feel up to the task, go for it.

Getting to use C++ is as easy as Python, all you need is the right set of tools in which to communicate with the computer in C++ and you can start your journey. A C++ IDE is free of charge, even the immensely powerful Visual Studio from Microsoft is freely available to download and use. You can get into C++ from any operating system, be it macOS, Linux, Windows or even mobile platforms.

Just like Python, to answer the question of Why C++ is the answer is because it's fast, efficient and developed by most of the applications you regularly use. It's cutting edge and a fantastic language to master.

Indeed, the operating system you're using is written in C++.

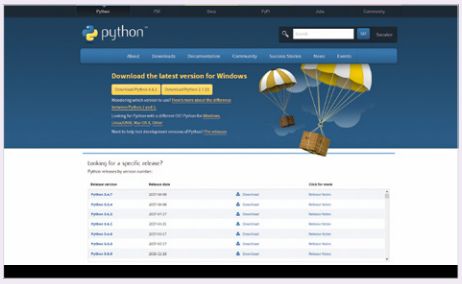
Windows 10 Mac debian ubuntu

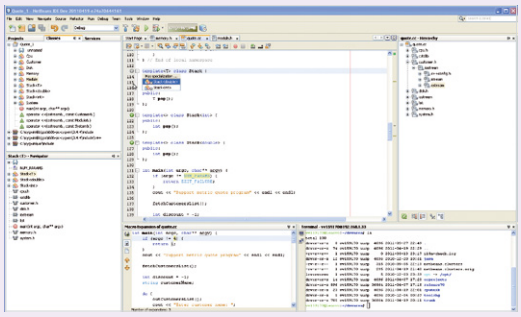
Equipment You Will Need

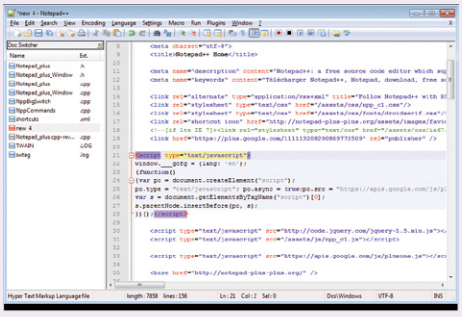
You don't need to invest a huge amount of money in order to learn C++ and you don't need an entire computing lab at your disposal either. Providing you have a fairly modern computer, everything else is freely available.

C++ SETUPS

Most, if not all, operating systems have C++ in their code, so it stands to reason that you can learn to program in C++ no matter what OS you're currently using.







COMPUTER

Unless you fancy writing out your C++ code by hand on a sheet of paper (which is something many older coders used to do), a computer is an absolute must have component. PC users can have any recent Linux distro or Windows OS, Mac users the latest macOS.

AN IDE

Just as with Python, an IDE is used to enter and execute your C++ code. Many IDEs come with extensions and plugins that help make it work better, or add an extra level of functionality. Often, an IDE provides enhancements depending on the core OS being used, such as being enhanced for Windows 10.

COMPILER

A compiler is a program that converts the C++ language into binary, so that the computer can understand. While some IDEs come with a compiler built in, others don't. Code::Blocks is our favourite IDE that comes with a C++ compiler as part of the package. More on this later.

TEXT EDITOR

Some programmers much prefer to use a text editor to assemble their C++ code before running it through a compiler. Essentially you can use any text editor to write code, just save it with a .cpp extension. However, Notepad++ is one of the best code text editors available.

INTERNET ACCESS

While it's entirely possible to learn how to code on a computer that's not attached to the Internet, it's extraordinarily difficult. You need to install relevant software, keep it up to date, install any extras or extensions and look for help when coding. All of these require access to the Internet.

TIME AND PATIENCE

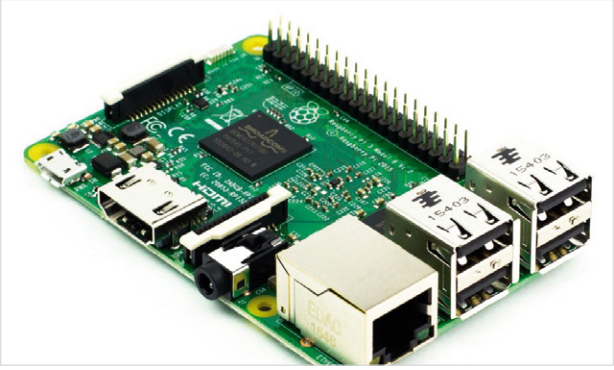
Yes, as with Python, you're going to need to set aside significant time to spend on learning how to code in C++. Sadly, unless you're a genius, it's not going to happen overnight, or even a week. A good C++ coder has spent many years honing their craft, so be patient, start small and keep learning.

OS SPECIFIC NEEDS

C++ will work in any operating system but getting all the necessary pieces together can be confusing to a newcomer. Here are some OS specifics for C++.

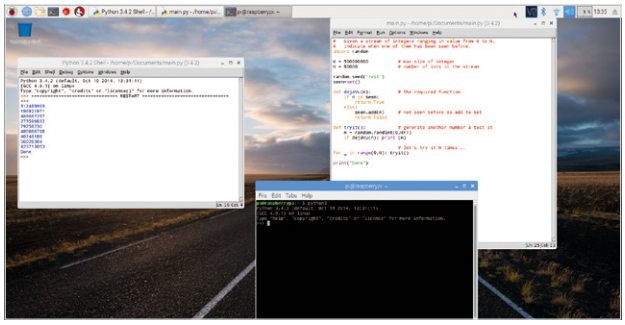
LINUX

Linux users are lucky in that they already have a compiler and text editor built into their operating system. Any text editor allows you to type out your C++ code, when it's saved with a .cpp extension, use g++ to compile it.



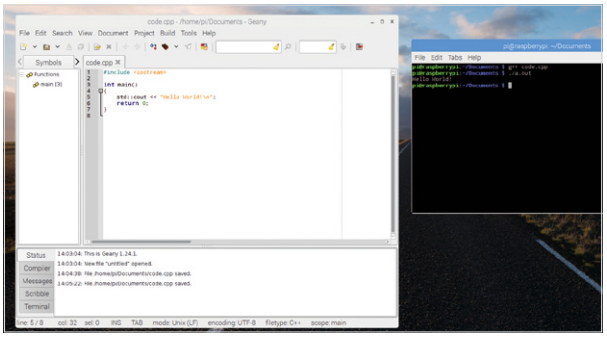
WINDOWS

We have mentioned previously that a good IDE is Microsoft's Visual Studio. However, a better IDE and compiler is Code::Blocks, which is regularly kept up to date with a new release twice a year. Otherwise Windows users can enter their code in Notepad++, then compile it with MinGW as used by Code::Blocks.



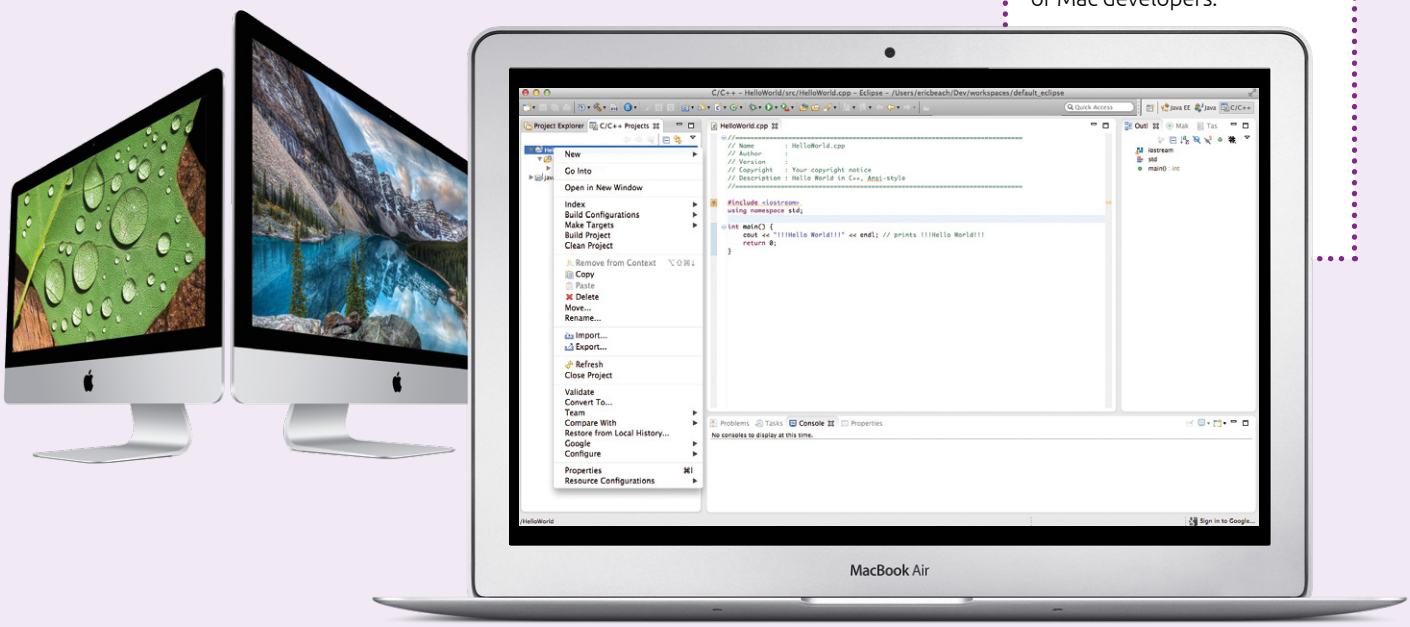
RASPBERRY PI

The Raspberry Pi's operating system is Raspbian, which is Linux based. Therefore, you're able to write your code out using a text editor, then compile it with g++ as you would in any other Linux distro.



MAC

Mac owners will need to download and install Xcode to be able to compile their C++ code natively. Other options for the macOS include Netbeans, Eclipse or Code::Blocks. Note: the latest Code::Blocks isn't available for Mac due to a lack of Mac developers.



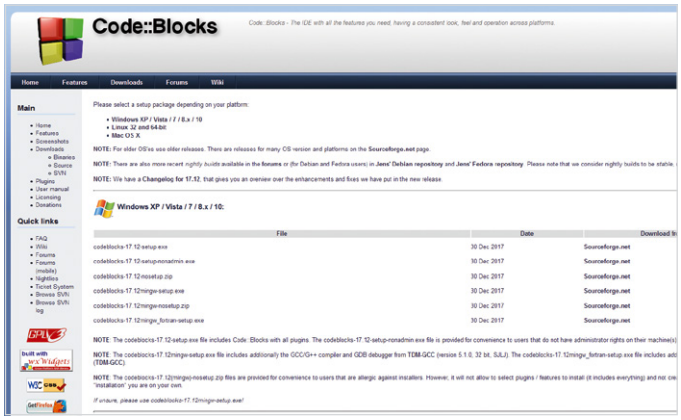
How to Set Up C++ in Windows

Windows users have a wealth of choice when it comes to programming in C++. There are plenty of IDEs and compilers available, including Visual Studio from Microsoft. However, in our opinion, the best C++ IDE to begin with is Code::Blocks.

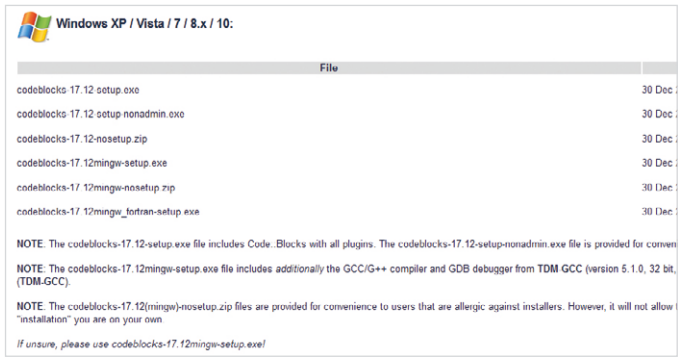
CODE::BLOCKS

Code::Blocks is a free C++, C and Fortran IDE that's feature rich and easily extendible with plug-ins. It's easy to use, comes with a compiler and has a vibrant community behind it.

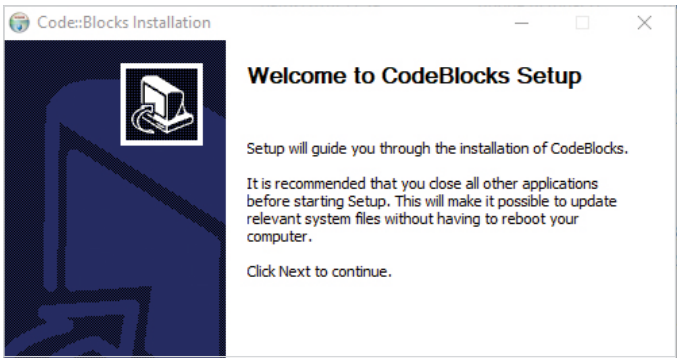
STEP 1 Start by visiting the Code::Blocks download site, at **www.codeblocks.org/downloads**. From there, click on the 'Download the binary releases' link to be taken to the latest downloadable version for Windows.



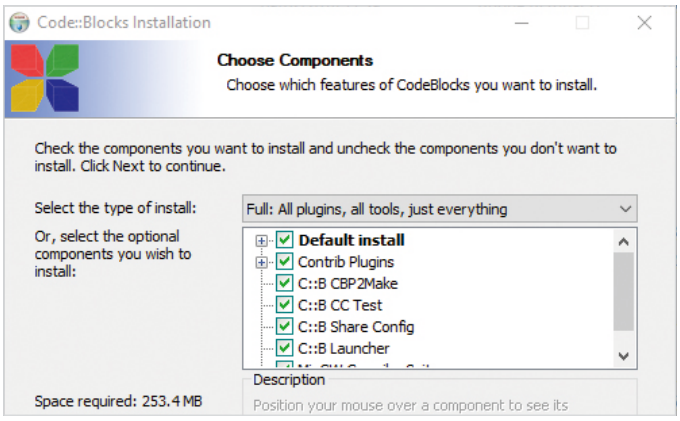
STEP 2 You can see that there are several Windows versions available. The one you want to download has 'mingw-setup.exe' at the end of the current version number. At the time of writing this is: codeblocks-17.12mingw-setup.exe. The difference is that the mingw-setup version includes a C++ compiler and debugger from TDM-GCC (a compiler suite).



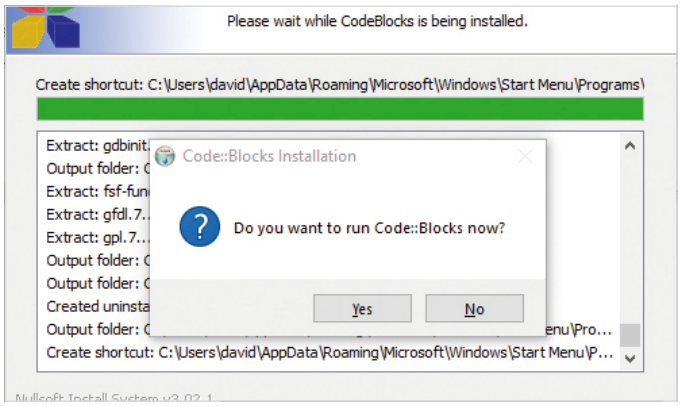
STEP 3 When you've located the file, click on the Sourceforge.net link at the end of the line and a download notification window appears; click on Save File to start the download and save the executable to your PC. Locate the downloaded Code::Blocks installer and double-click to start. Follow the on-screen instructions to begin the installation.



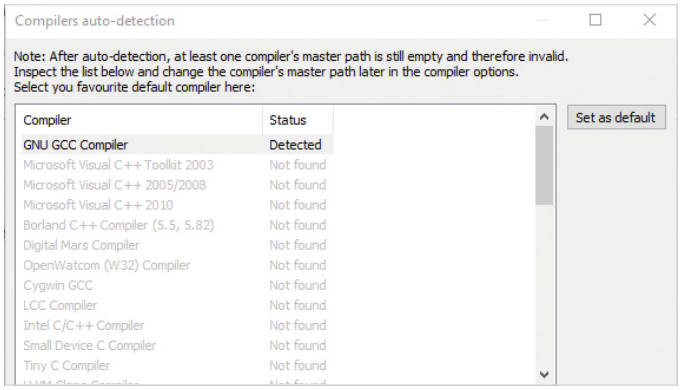
STEP 4 Once you've agreed to the licencing terms, there is a choice of installation options available. You can opt for a smaller install, missing out on some of the components but we would recommend you opt for the Full option as default.



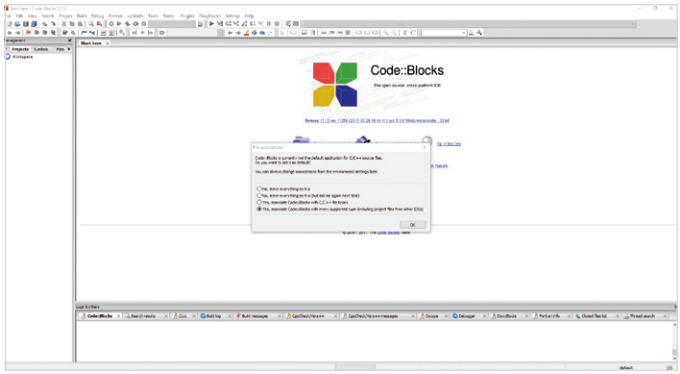
STEP 5 Next choose an install location for the Code::Blocks files. It's your choice, but the default will generally suffice, unless of course you have any special requirements. When you click Next, the install begins; when it's finished a notification pops up asking you if you want to start Code::Blocks now, so click Yes.



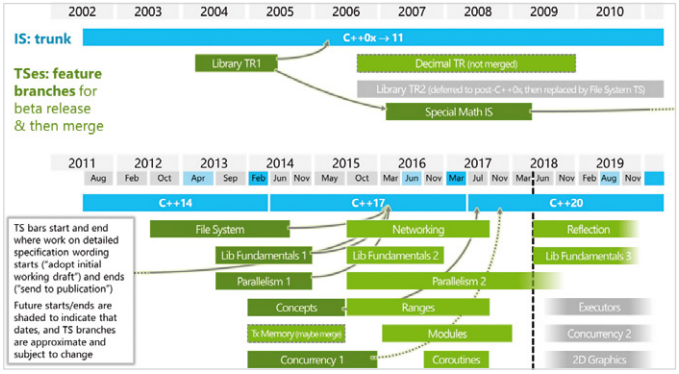
STEP 6 The first time Code::Blocks loads it runs an auto-detect for any C++ compilers you may already have installed on you system. If you don't have any, click on the first detected option, GNU GCC Compiler, and click the Default button to set it as the system's C++ compiler. Click OK when you're ready to continue.



STEP 7 When the program starts another message appears, informing you that Code::Blocks is currently not the default application for C++ files. You a couple of options: to leave everything as it is or allow Code::Blocks to associate all C++ file types. Again, we would recommend you opt for the last choice to associate Code::Blocks with every supported file type.

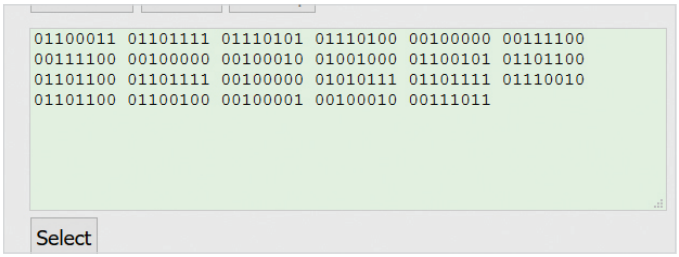


STEP 8 Before you start using Code::Blocks it's worth explaining exactly why you need the added compiler. First, a compiler is a separate program that reads through your C++ code and checks it against the latest acceptable programming standards; this is why you need the most recently available compiler. This is currently C++17, with C++20 underway.

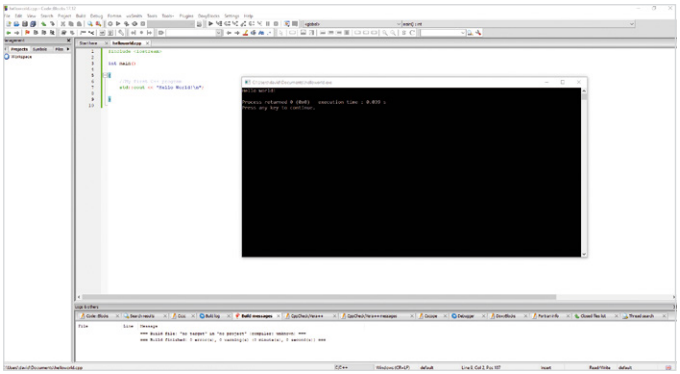


STEP 9 Essentially, computers work and understand only binary, ones and zeros, or Machine Language. Programming in binary isn't effective for human beings. For example, to output the words "Hello World!" to the screen in C++ would appear in binary as:

```
01100011 01101111 01110101 01110100 00100000
00111100 00111100 00100000 00100010 01001000
01100101 01101100 01101100 01101111 00100000
01010111 01101111 01110010 01101100 01100100
00100001 00100010 00111011
```



STEP 10 The compiler therefore takes what you've entered as C++ code and translates that to Machine Language. To execute C++ code the IDE 'builds' the code, checking for errors, then pass it through the compiler to check standardisation and convert it to ones and zeros for the computer to act upon. It's rather clever stuff, when you stop to think about it.



How to Set Up C++ on a Mac

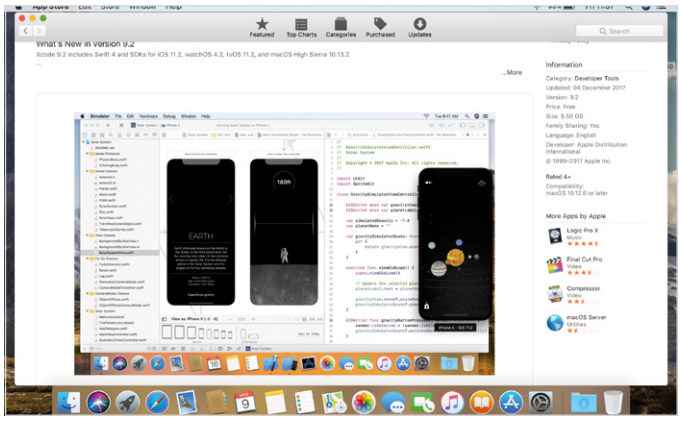
To begin C++ coding on a Mac you first need to install Apple's Xcode. This is a free, full featured IDE that's designed to create native Apple apps. However, you can also use it to create C++ code relatively easily.

XCODE

Apple's Xcode is primarily designed for users to develop apps for macOS, iOS, tvOS and watchOS applications in Swift or Objective-C but you can use it for C++ too.

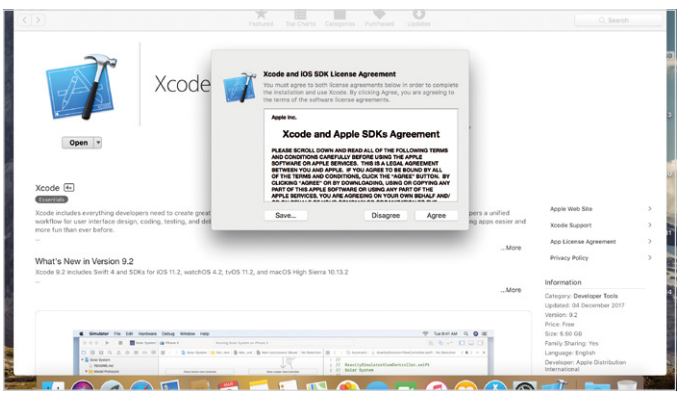
STEP 1 Start by opening the App Store on your Mac, Apple Menu > App Store. In the Search box enter 'Xcode' and press Return. There are many suggestions filling the App Store window but it's the first option, Xcode, that you need to click on.

STEP 2 Take a moment to browse through the app's information, including the compatibility, to ensure you have the correct version of macOS. Xcode requires macOS 10.12.6 or later to install and work.



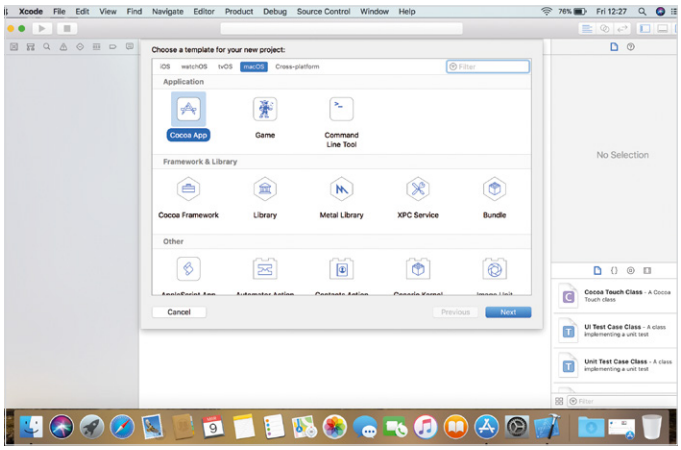
STEP 3 When you're ready, click on the Get button which then turns into Install App. Enter your Apple ID and Xcode begins to download and install. It may take some time depending on the speed of your Internet connection.

STEP 4 When the installation is complete, click on the Open button to launch Xcode. Click Agree to the licence terms and enter your password to allow Xcode to make changes to the system. When that is done, Xcode begins to install additional components.

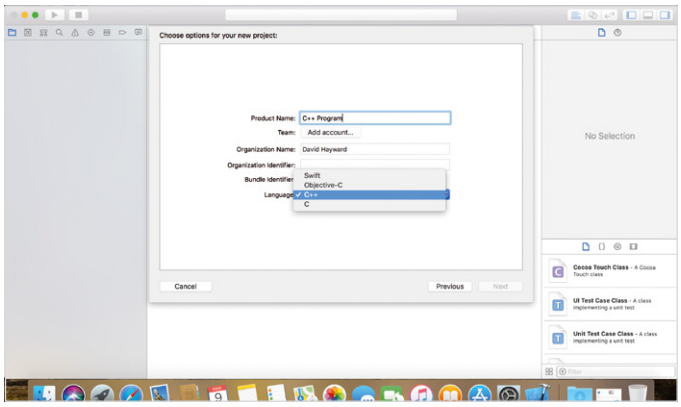


STEP 5 With everything now installed, including the additional components, Xcode launches, displaying the version number along with three choices and any recent projects that you've worked on; with a fresh install though, this is blank.

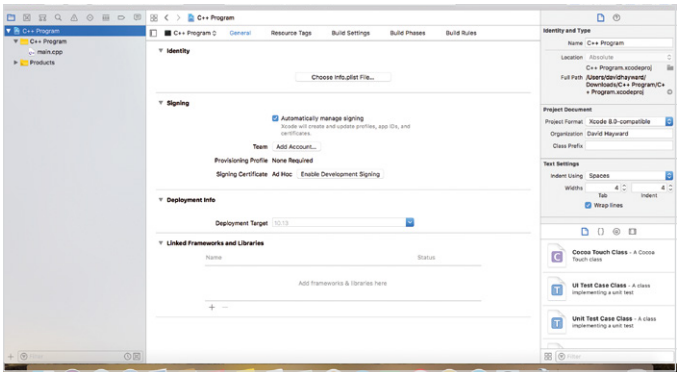
STEP 6 Start by clicking on Create New Xcode Project; this opens a template window to choose which platform you're developing code for. Click the macOS tab, then click the Command Line Tool option. Click Next to continue.



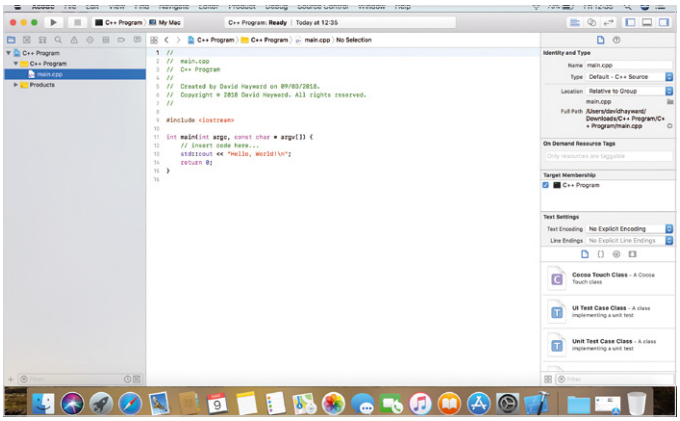
STEP 7 Fill in the various fields but ensure that the Language option at the bottom is set to C++; simply choose it from the drop-down list. When you've filled in the fields, and made sure that C++ is the chosen language, click on the Next button to continue.



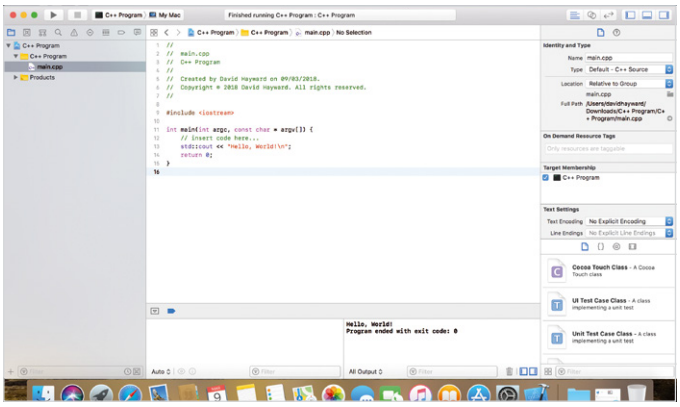
STEP 8 The next step asks where to create a Git Repository for all your future code. Choose a location on your Mac, or a network location, and click the Create button. When you've done all that, you can start to code. The left-hand pane details the files used in the C++ program you're coding. Click on the main.cpp file in the list.



STEP 9 You can see that Xcode has automatically completed a basic Hello World program for you. While it may not make much sense at present, you will discover more as you progress, the content is just Xcode utilising what's available on the Mac.



STEP 10 When you want to run the code, click on Product > Run. You may be asked to enable Developer Mode on the Mac; this is to authorise Xcode to perform functions without needing your password every session. When the program executes, the output is displayed at the bottom of the Xcode window.



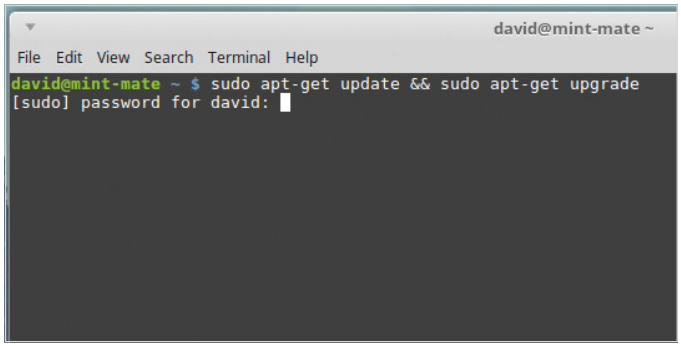
How to Set Up C++ in Linux

Linux is a great C++ coding environment. Most Linux distros already have the essential components preinstalled, such as a compiler. The text editors are also excellent for entering code into, including colour coding. There's also tons of extra software available to help you out.

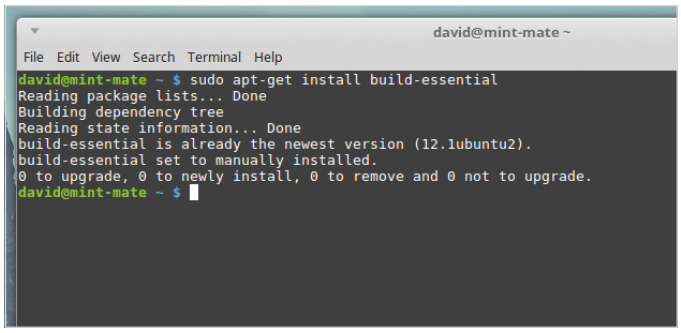
LINUX++

If you're not familiar with Linux, then we recommend taking a look at one of our Linux titles from the BDM Publications range. If you have a Raspberry Pi, the commands used below work just fine and for this example we're using Linux Mint.

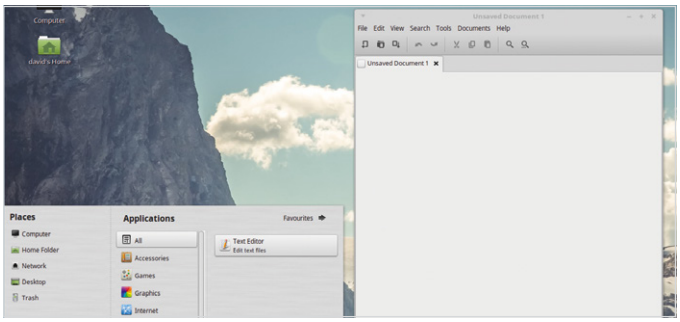
STEP 1 The first step is to ensure Linux is ready for your C++ code, so check the system and software are up to date. Open a Terminal and enter: **sudo apt-get update && sudo apt-get upgrade**. Then press Return and enter your password. These commands update the entire system and any installed software.



STEP 2 Most Linux distros come preinstalled with all the necessary components to start coding in C++; however, it's always worth checking to see if everything is present. Still within the Terminal, enter: s**udo apt-get install build-essential** and press Return. If you have the right components nothing is installed; if you're missing some then they are installed by the command.

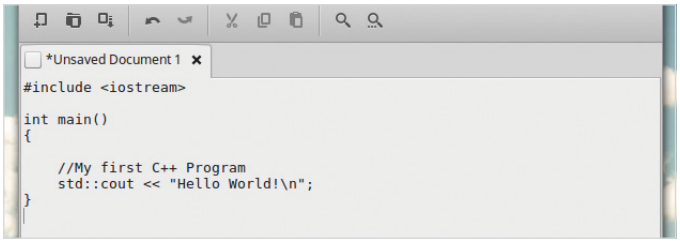


STEP 3 Amazingly, that's it. Everything is already for you to start coding. Here's how to get your first C++ program up and running. In Linux Mint the main text editor is Xed, which you can launch by clicking on the Menu and typing Xed into the search bar. Click on the Text Editor button in the right-hand pane to open it.

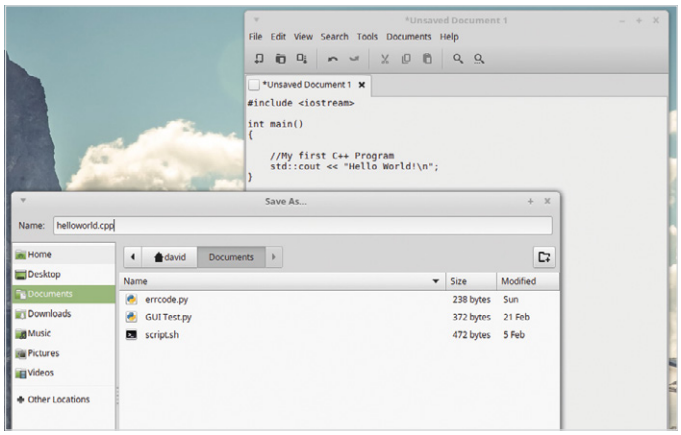


STEP 4 In Xed, or any other text editor you may be using, enter the lines of code that make up your C++ Hello World program. It's a little different to what the Mac produced:

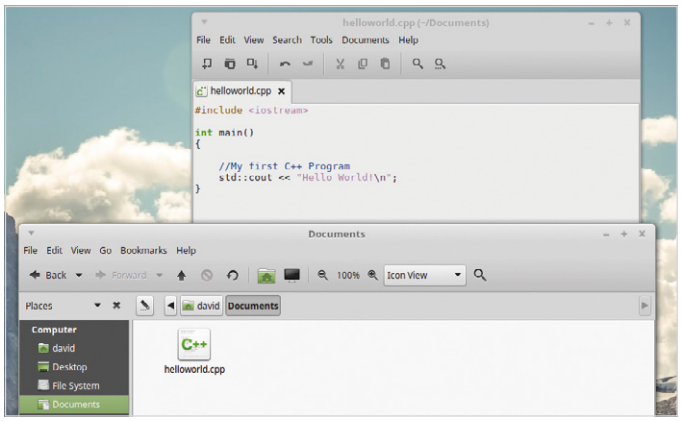
```
#include <iostream>

int main()
{
//My first C++ program
std::cout <<  "Hello World!\n";
}
```



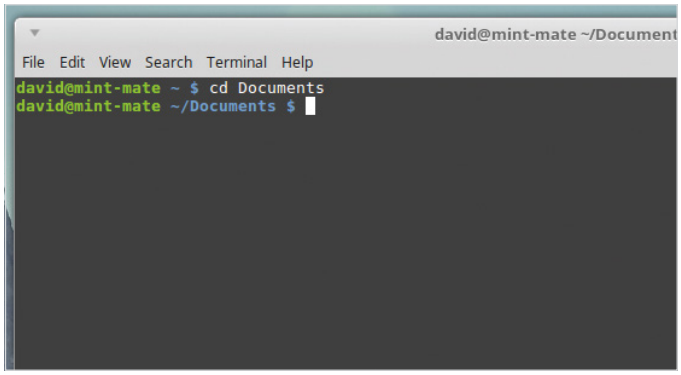
STEP 5 When you've entered your code, click File > Save As and choose a folder in which to save your program. Name the file as helloworld.cpp (it can be any name as long as it has .cpp as the extension). Click Save to continue.



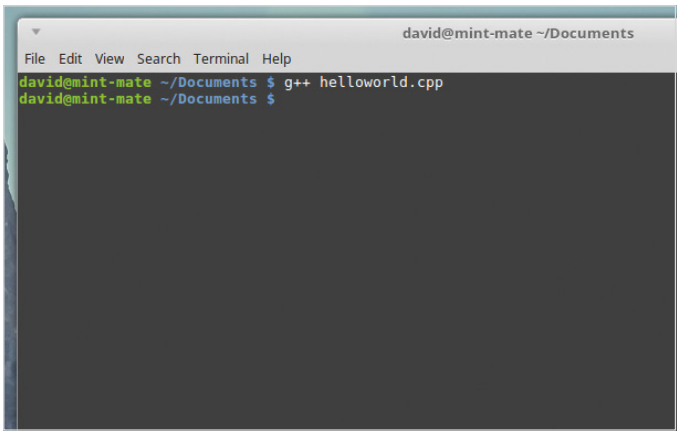
STEP 6 The first thing to notice is that Xed has automatically recognised this as a C++ file, since the file extension is now set to .cpp. The colour coding is present in the code and if you open up the file manager you can also see that file's icon has C++ stamped on it.



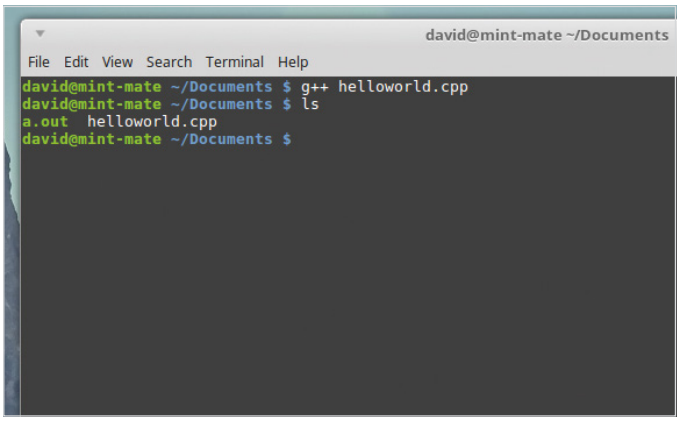
STEP 7 With your code now saved, drop into the Terminal again. You need to navigate to the location of the C++ file you've just saved. Our example is in the Documents folder, so we can navigate to it by entering: **cd Documents**. Remember, the Linux Terminal is case sensitive, so any capitals must be entered correctly.



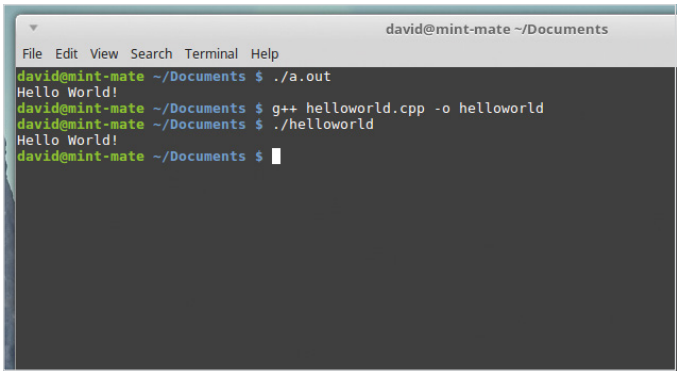
STEP 8 Before you can execute the C++ file you need to compile it. In Linux it's common to use g++, an open source C++ compiler; as you're now in the same folder as the C++ file, enter: **g++ helloworld.cpp** in the Terminal and press Return.



STEP 9 It takes a short time while the code is compiled by g++ but providing there are no mistakes or errors in the code you are returned to the command prompt. The compiling of the code has created a new file. If you enter: ls into the Terminal you can see that alongside your C++ file is a.out.



STEP 10 The a.out file is the compiled C++ code. To run the code enter: **./a.out** and press Return. The words 'Hello World!' appear on the screen. However, a.out isn't very friendly. To name it something else post-compiling, you can recompile with: **g++ helloworld.cpp -o helloworld**. This creates an output file called helloworld which can be run with: **./helloworld**.



Other C++ IDEs to Install

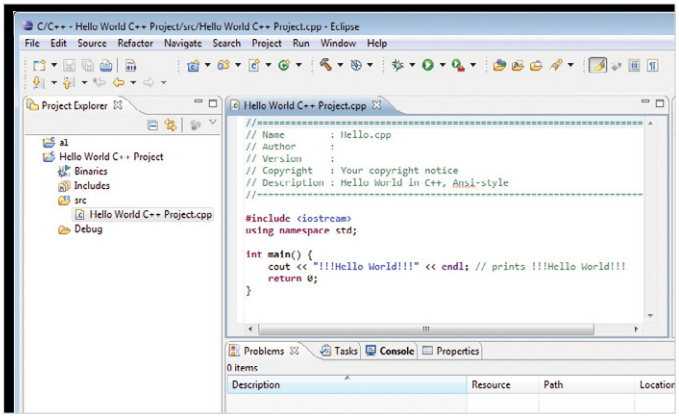
If you want to try a different approach to working with your C++ code, then there are plenty of options available to you. Windows is the most prolific platform for C++ IDEs but there are plenty for Mac and Linux users too.

DEVELOPING C++

Here are ten great C++ IDEs that are worth looking into. You can install one or all of them if you like, but find the one that works best for you.

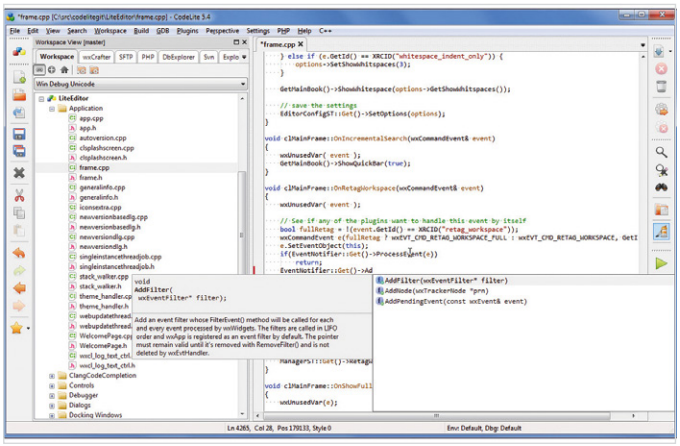
ECLIPSE

Eclipse is a hugely popular C++ IDE that offers the programmer a wealth of features. It has a great, clean interface, is easy to use and available for Windows, Linux and Mac. Head over to **www.eclipse.org/downloads/** to download the latest version. If you're stuck, click the Need Help link for more information.



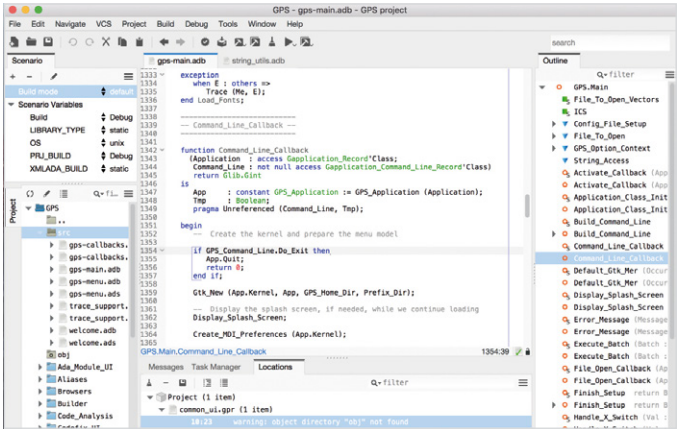
CODELITE

CodeLite is a free and open source IDE that's regularly updated and available for Windows, Linux and macOS. It's lightweight, uncomplicated and extremely powerful. You can find out more information as well as how to download and install it at **www.codelite.org/**.



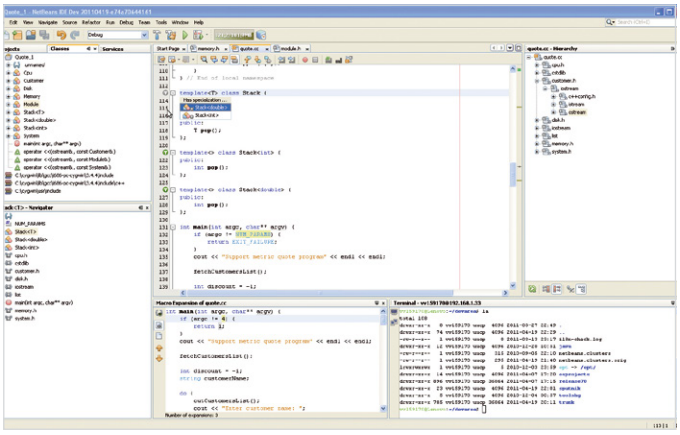
GNAT

The GNAT Programming Studio (GPS) is a powerful and intuitive IDE that supports testing, debugging and code analysis. The Community Edition is free, whereas the Pro version costs; however, the Community Edition is available for Windows, Mac, Linux and even the Raspberry Pi. You can find it at **www.adacore.com/download**.



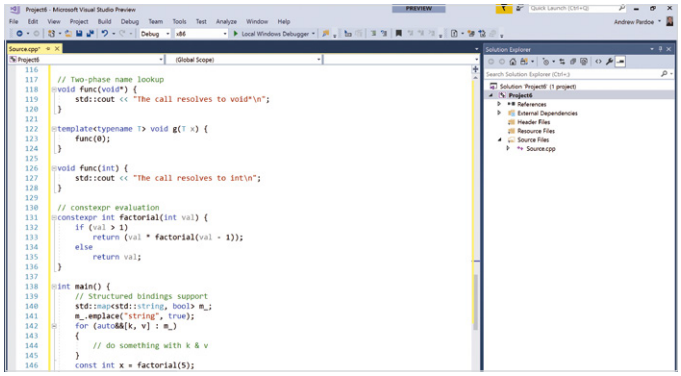
NETBEANS

Another popular choice is NetBeans. This is another excellent IDE that's packed with features and a pleasure to use. NetBeans IDE includes project based templates for C++ that give you the ability to build applications with dynamic and static libraries. Find out more at **www.netbeans.org/features/cpp/index.html.**



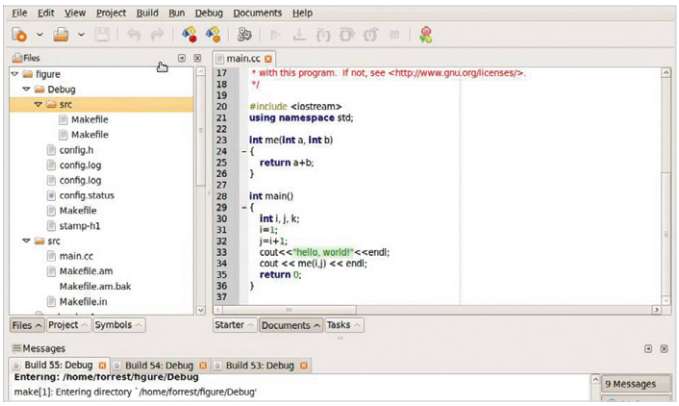
VISUAL STUDIO

Microsoft's Visual Studio is a mammoth C++ IDE that allows you to create applications for Windows, Android, iOS and the web. The Community version is free to download and install but the other versions allow a free trial period. Go to **www.visualstudio.com/** to see what it can do for you.



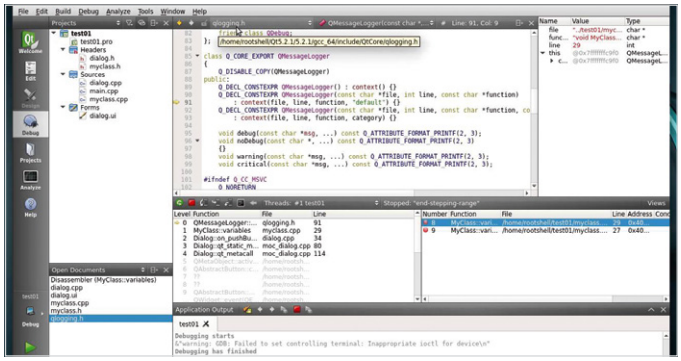
ANJUTA

The Anjuta DevStudio is a Linux-only IDE that features some of the more advanced features you would normally find in a paid software development studio. There's a GUI designer, source editor, app wizard, interactive debugger and much more. Go to **www.anjuta.org/** for more information.



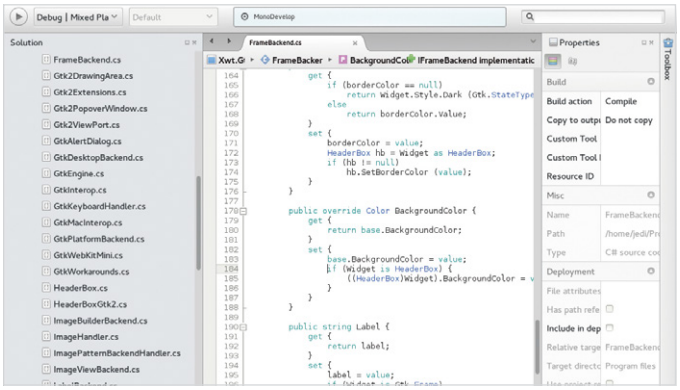
QT CREATOR

This cross-platform IDE is designed to create C++ applications for desktop and mobile environments. It comes with a code editor and integrated tools for testing and debugging, as well as deploying to you chosen platform. It's not free but there is a trial period on offer before requiring purchasing: **www.qt.io/qt-features-libraries-apis-tools-and-ide/**.



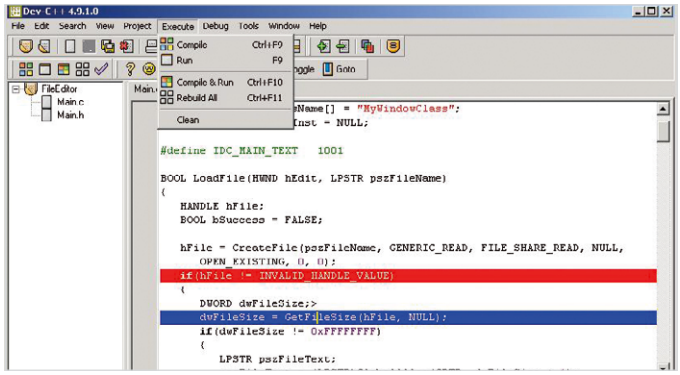
MONODEVELOP

This excellent IDE allows developers to write C++ code for desktop and web applications across all the major platforms. There's an advanced text editor, integrated debugger and a configurable workbench to help you create your code. It's available for Windows, Mac and Linux and is free to download and use: **www.monodevelop.com/**.



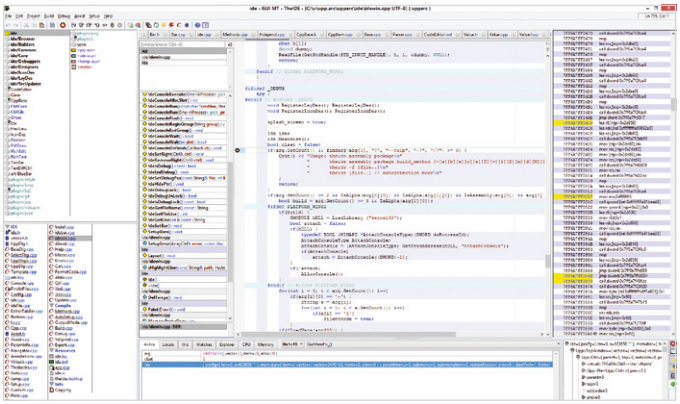
DEV C++

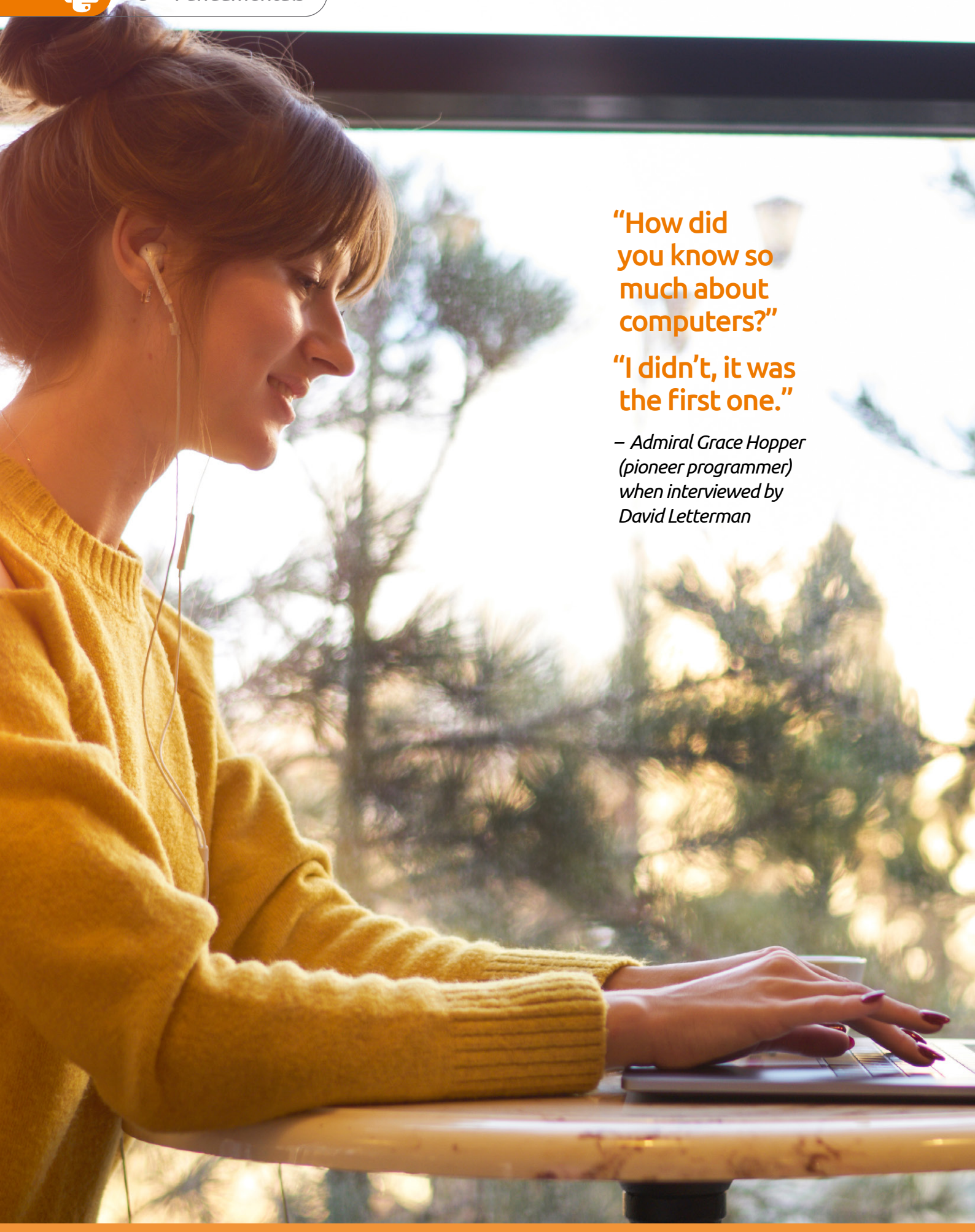
Bloodshed Dev C++, despite its colourful name, is an older IDE that is for Windows systems only. However, many users praise its clean interface and uncomplicated way of coding and compiling. Although there's not been much updating for some time, it's certainly one to consider if you want something different: **www.bloodshed.net/devcpp.html**.



U++

Ultimate++ is a cross-platform C++ IDE that boasts a rapid development of code through the smart and aggressive use of C++. For the novice, it's a beast of an IDE but behind its complexity is a beauty that would make a developer's knees go wobbly. Find out more at **www.ultimatepp.org/index.html**.





"How did you know so much about computers?"

"I didn't, it was the first one."

– Admiral Grace Hopper (pioneer programmer) when interviewed by David Letterman

C++ Fundamentals

Within this section, you will begin to understand the structure of C++ code and how you can compile and execute that code. These are the core fundamentals of C++, which teach you the basics such as using comments, variables, data types, strings and how to use C++ mathematics.

These are the building blocks of a C++ program. With them, you will be able to form your own code, produce an output to the screen, store and retrieve data and be well on your way to creating your own custom code.

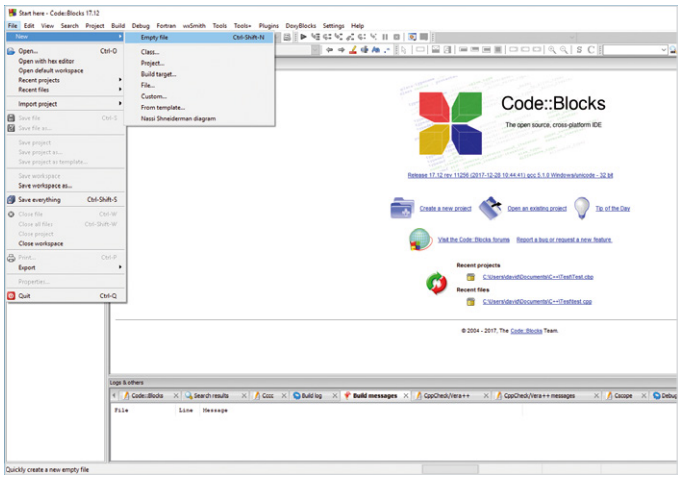
Your First C++ Program

You may have followed the Mac and Linux examples previously but you're going to be working exclusively in Windows and Code::Blocks from here on. Let's begin by writing your first C++ program and taking the first small step into a larger coding world.

HELLO, WORLD!

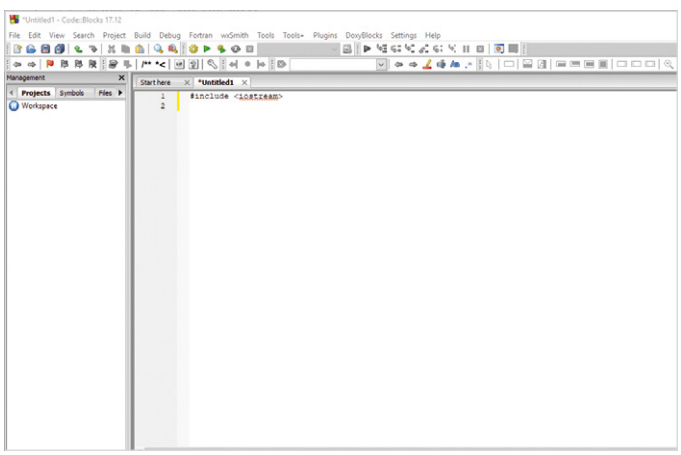
It's traditional in programming for the first code to be entered to output the words 'Hello, World!' to the screen. Interestingly, this dates back to 1968 using a language called BCPL.

STEP 1 As mentioned, we're using Windows 10 and the latest version of Code::Blocks for the rest of the C++ code in this book. Begin by launching Code::Blocks. When open, click on File > New > Empty File or press Ctrl+Shift+N on the keyboard.

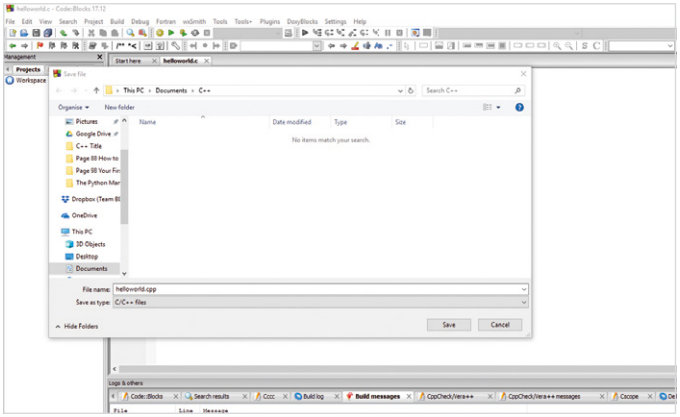


STEP 2 Now you can see a blank screen, with the tab labelled *Untitled1, and the number one in the top left of the main Code::Blocks window. Begin by clicking in the main window, so the cursor is next to the number one, and entering:

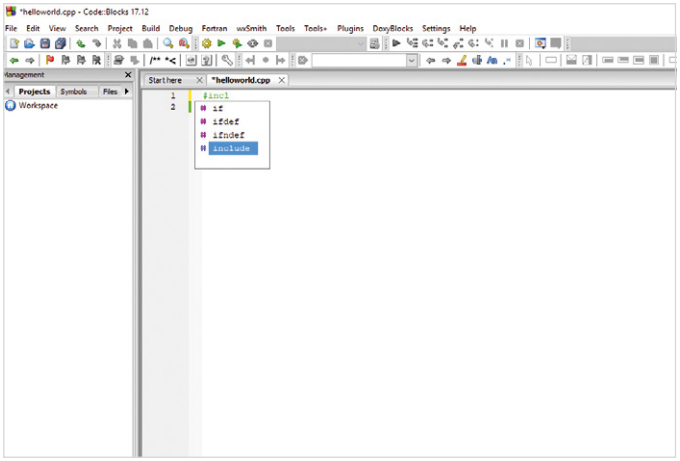
```
#include <iostream>
```



STEP 3 At the moment it doesn't look like much, and it makes even less sense, but we'll get to that in due course. Now click on File > Save File As. Create or find a suitable location on your hard drive and in the File Name box, call it helloworld.cpp. Click the Save as type box and select C/C++ files. Click the Save button.



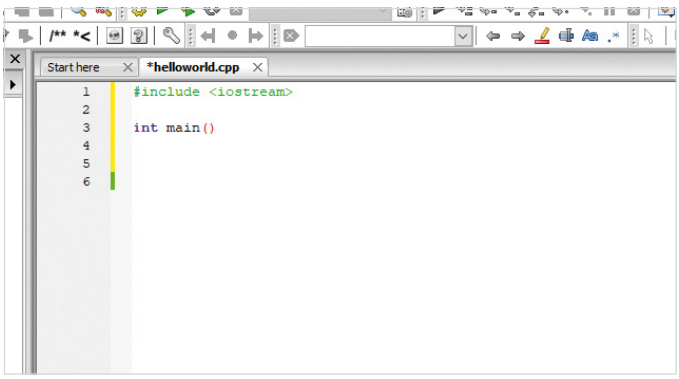
STEP 4 You can see that Code::Blocks has now changed the colour coding, recognising that the file is now C++ code. This means that code can be auto-selected from the Code::Blocks repository. Delete the #include <iostream> line and re-enter it. You can see the auto-select boxes appearing.



STEP 5 Auto-selection of commands is extremely handy and cuts out potential mistyping. Press Return to get to line 3, then enter:

```
int main()
```

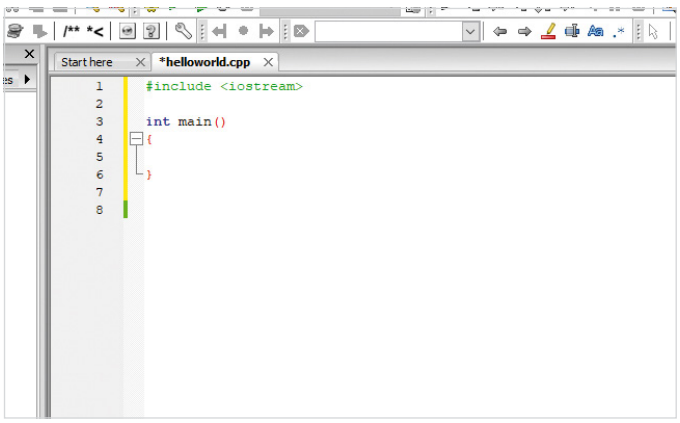
Note: there's no space between the brackets.



STEP 6 On the next line below int main(), enter a curly bracket:

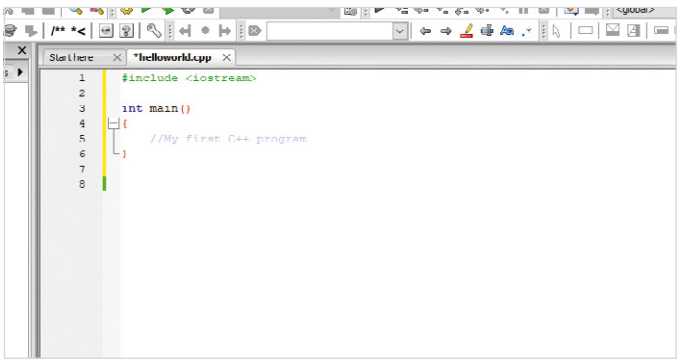
```
{
```

This can be done by pressing Shift and the key to the right of P on an English UK keyboard layout.



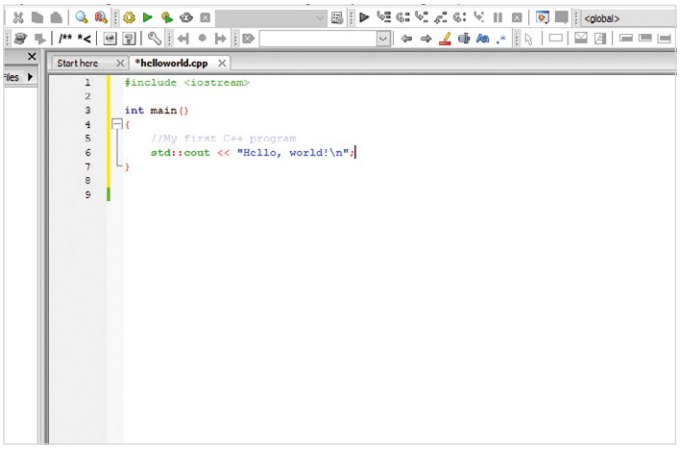
STEP 7 Notice that Code::Blocks has automatically created a corresponding closing curly bracket a couple of lines below, linking the pair, as well as a slight indent. This is due to the structure of C++ and it's where the meat of the code is entered. Now enter:

```
//My first C++ program
```



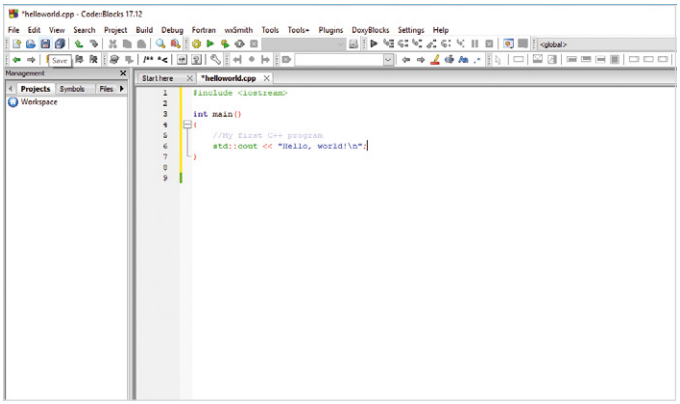
STEP 8 Note again the colour coding change. Press Return at the end of the previous step's line, and then enter:

```
std::cout << "Hello, world!\n";
```



STEP 9 Just as before, Code::Blocks auto-completes the code you're entering, including placing a closing speech mark as soon as you enter the first. Don't forget the semicolon at the end of the line; this is one of the most important elements to a C++ program and we'll tell you why in the next section. For now, move the cursor down to the closing curly bracket and press Return.

STEP 10 That's all you need to do for the moment. It may not look terribly amazing but C++ is best absorbed in small chunks. Don't execute the code at the moment as you need to look at how a C++ program is structured first; then you can build and run the code. For now, click on Save, the single floppy disc icon.



Structure of a C++ Program

C++ has a very defined structure and way of doing things. Miss something out, even as small as a semicolon, and your entire program will fail to be compiled and executed. Many a professional programmer has fallen foul of sloppy structure.

#INCLUDE <C++ STRUCTURE>

Learning the basics of programming, you begin to understand the structure of a program. The commands may be different from one language to the next, but you will start to see how the code works.

C++

C++ was invented by Danish student Bjarne Stroustrup in 1979, as a part of his Ph.D. thesis. Initially C++ was called C with Classes, which added features to the already popular C programming language, while making it a more user-friendly environment through a new structure.

Bjarne Stroustrup, inventor of C++.

#INCLUDE

The structure of a C++ program is quite precise. Every C++ code begins with a directive: **#include <>**. The directive instructs the pre-processor to include a section of the standard C++ code. For example: **#include <iostream>** includes the iostream header to support input/output operations.

Start here | *helloworld.cpp

```
1  #include <iostream>
2
3
4
5
6
```

INT MAIN()

int main() initiates the declaration of a function, which is a group of code statements under the name 'main'. All C++ code begins at the main function, regardless of where it actually lies within the code.

Start here | *helloworld.cpp

```
1  #include <iostream>
2
3  int main()
4
5
6
```

BRACES

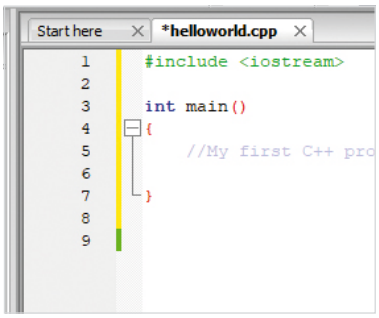
The open brace (curly brackets) is something that you may not have come across before, especially if you're used to Python. The open brace indicates the beginning of the main function and contains all the code that belongs to that function.

Start here | *helloworld.cpp

```
1  #include <iostream>
2
3  int main()
4  {
5
6  }
```

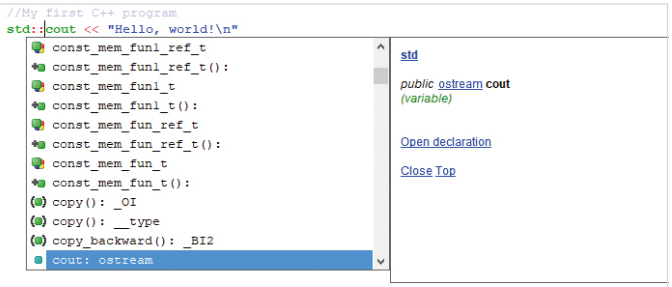
COMMENTS

Lines that begin with a double slash are comments. This means they won't be executed in the code and are ignored by the compiler. Comments are designed to help you, or another programmer looking at your code, explain what's going on. There are two types of comment: /* covers multiple line comments, // a single line. Lines that begin with a double slash are comments. This means they won't be executed in the code and are ignored by the compiler. Comments are designed to help you, or another programmer looking at your code, explain what's going on. There are two types of comment: /* covers multiple line comments, // a single line.



<<

The two chevrons used here are insertion operators. This means that whatever follows the chevrons is to be inserted into the std::cout statement. In this case they're the words 'Hello, world', which are to be displayed on the screen when you compile and execute the code.



STD

While **std** stands for something quite different, in C++ it means Standard. It's part of the Standard Namespace in C++, which covers a number of different statements and commands. You can leave the **std::** part out of the code but it must be declared at the start with: **using namespace std**; not both. For example:

```
#include <iostream>
using namespace std;
```

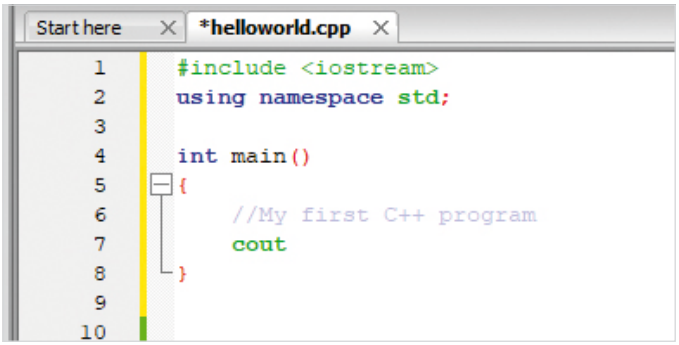
OUTPUTS

Leading on, the "Hello, world!" part is what we want to appear on the screen when the code is executed. You can enter whatever you like, as long as it's inside the quotation marks. The brackets aren't needed but some compilers insist on them. The \n part indicates a new line is to be inserted.

```
//My first C++ program
cout << "Hello, world!\n"
```

COUT

In this example we're using cout, which is a part of the Standard Namespace, hence why it's there, as you're asking C++ to use it from that particular namespace. Cout means Character OUTput, which displays, or prints, something to the screen. If we leave **std::** out we have to declare it at the start of the code, as mentioned previously.



; AND }

Finally you can see that lines within a function code block (except comments) end with a semicolon. This marks the end of the statement and all statements in C++ must have one at the end or the compiler fails to build the code. The very last line has the closing brace to indicate the end of the main function.

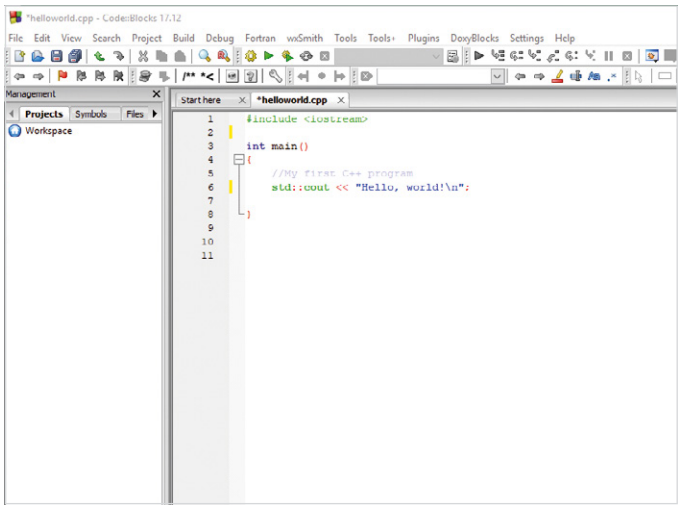
Compile and Execute

You've created your first C++ program and you now understand the basics behind the structure of one. Let's actually get things moving and compile and execute, or run if you prefer, the program and see how it looks.

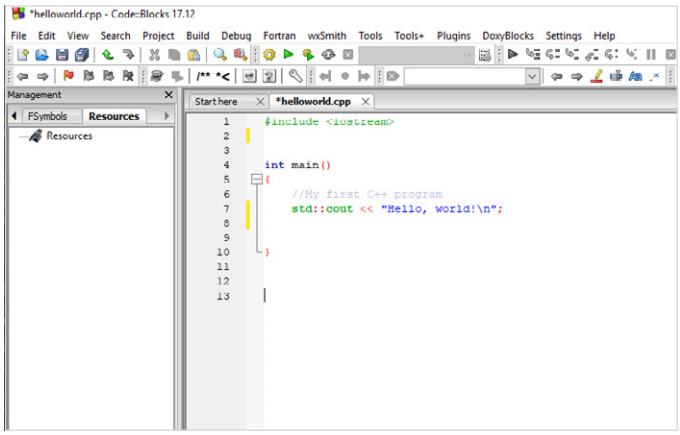
GREETINGS FROM C++

Compiling and executing C++ code from Code::Blocks is extraordinarily easy; just a matter of clicking an icon and seeing the result. Here's how it's done.

STEP 1 Open Code::Blocks, if you haven't already, and load up the previously saved Hello World code you created. Ensure that there are no visible errors, such as missing semicolons at the end of the std::cout line.

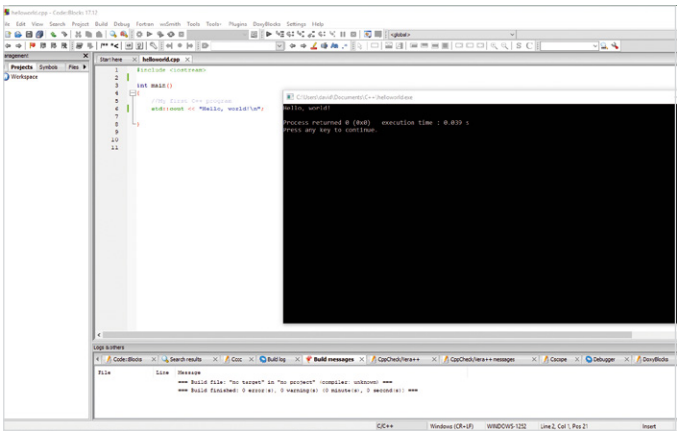


STEP 2 If your code is looking similar to the one in our screenshot, then look to the menu bar along the top of the screen. Under the Fortran entry in the topmost menu you can see a group of icons: a yellow cog, green play button and a cog/play button together. These are Build, Run, Build and Run functions.



STEP 3 Start by clicking on the Build icon, the yellow cog. At this point, your code has now been run through the Code::Blocks compiler and checked for any errors. You can see the results of the Build by looking to the bottom window pane. Any messages regarding the quality of the code are displayed here.

STEP 4 Now click on the Run icon, the green play button. A command line box appears on your screen displaying the words: Hello, world!, followed by the time it's taken to execute the code, and asking you press a key to continue. Well done, you just compiled and executed your first C++ program.



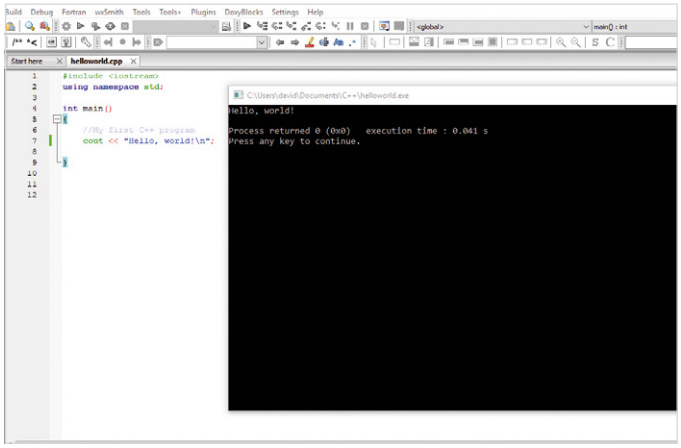
STEP 5 Pressing any key in the command line box closes it, returning you to Code::Blocks. Let's alter the code slightly. Under the #include line, enter:

```
using namespace std;
```

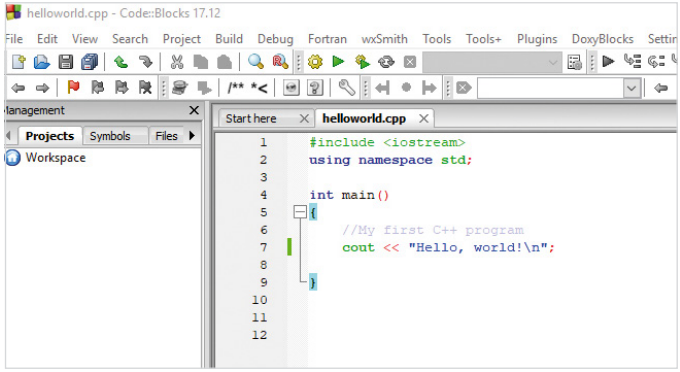
Then, delete the std:: part of the Cout line; like so:

```
cout << "Hello, world\n";
```

STEP 6 In order to apply the new changes to the code, you need to re-compile, build, and run it again. This time, however, you can simply click the Build/Run icon, the combined yellow cog and green play button.



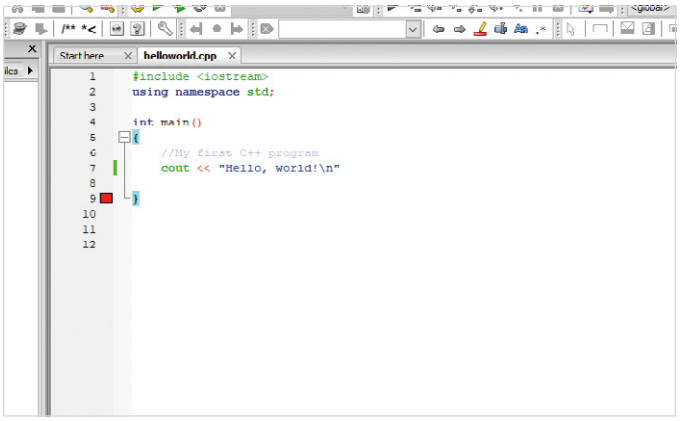
STEP 7 Just as we mentioned in the previous pages, you don't need to have std::cout if you already declare using namespace std; at the beginning of the code. We could have easily clicked the Build/Run icon to begin with but it's worth going through the available options. You can also see that by building and running, the file has been saved.



STEP 8 Create a deliberate error in the code. Remove the semicolon from the cout line, so it reads:

```
cout << "Hello, world!\n"
```

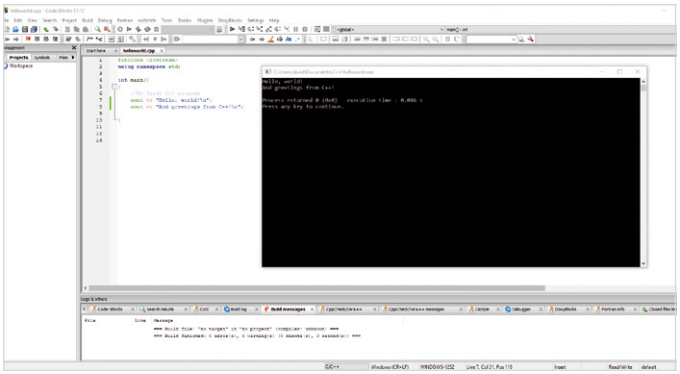
STEP 9 Now click the Build and Run icon again to apply the changes to the code. This time Code::Blocks refuses to execute the code, due to the error you put in. In the Log pane at the bottom of the screen you are informed of the error, in this case: Expected ';' before '}' token, indicating the missing semicolon.



STEP 10 Replace the semicolon and under the cout line, enter a new line to your code:

```
cout << "And greetings from C++!\n";
```

The \n simply adds a new line under the last line of outputted text. Build and Run the code, to display your handiwork.



Using Comments

While comments may seem like a minor element to the many lines of code that combine to make a game, application or even an entire operating system, in actual fact they're probably one of the most important factors.

THE IMPORTANCE OF COMMENTING

Comments inside code are basically human readable descriptions that detail what the code is doing at that particular point. They don't sound especially important but code without comments is one of the many frustrating areas of programming, regardless of whether you're a professional or just starting out.

In short, all code should be commented in such a manner as to effectively describe the purpose of a line, section, or individual elements. You should get in to the habit of commenting as much as possible, by imagining that someone who doesn't know anything about programming can pick up your code and understand what it's going to do simply by reading your comments.

In a professional environment, comments are vital to the success of the code and ultimately, the company. In an organisation, many programmers work in teams alongside engineers, other developers, hardware analysts and so on. If you're a part of the team that's writing a bespoke piece of software for the company, then your comments help save a lot of time should something go wrong, and another team member has to pick up and follow the trail to pinpoint the issue.

Place yourself in the shoes of someone whose job it is to find out what's wrong with a program. The program has in excess of 800,000 lines of code, spread across several different modules. You can soon appreciate the need for a little help from the original programmers in the form of a good comment.

The best comments are always concise and link the code logically, detailing what happens when the program hits this line or section. You don't need to comment on every line. Something along the lines of: if x==0 doesn't require you to comment that if x equals zero then do something; that's going to be obvious to the reader. However, if x equalling zero is something that drastically changes the program for the user, such as, they've run out of lives, then it certainly needs to commented on.

Even if the code is your own, you should write comments as if you were going to publicly share it with others. This way you can return to that code and always understand what it was you did or where it was you went wrong or what worked brilliantly.

Comments are good practise and once you understand how to add a comment where needed, you soon do it as if it's second nature.

```
        DEFB    26h,30h,32h,26h,30h,32h,0,0,32h,72h,73h,32h,72h,73h,32h
        DEFB    60h,61h,32h,4Ch,4Dh,32h,4Ch,99h,32h,4Ch,4Dh,32h,4Ch,4Dh
        DEFB    32h,4Ch,99h,32h,5Bh,5Ch,32h,56h,57h,32h,33h,0CDh,32h,33h
        DEFB    34h,32h,33h,34h,32h,33h,0CDh,32h,40h,41h,32h,66h,67h,64h
        DEFB    66h,67h,32h,72h,73h,64h,4Ch,4Dh,32h,56h,57h,32h,80h,0CBh
        DEFB    19h,80h,0,19h,80h,81h,32h,80h,0CBh,0FFh
T858C:
        DEFB    80h,72h,66h,60h,56h,66h,56h,56h,51h,60h,51h,51h,56h,66h
        DEFB    56h,56h,80h,72h,66h,60h,56h,66h,56h,56h,51h,60h,51h,51h
        DEFB    56h,56h,56h,56h,80h,72h,66h,60h,56h,66h,56h,56h,51h,60h
        DEFB    51h,51h,56h,66h,56h,56h,80h,72h,66h,60h,56h,66h,56h,40h
        DEFB    56h,66h,80h,66h,56h,56h,56h,56h
;
; Game restart point
;
START:  XOR     A
        LD      (SHEET),A
        LD      (KEMP),A
        LD      (DEMO),A
        LD      (B845B),A
        LD      (B8458),A
        LD      A,2             ;Initial lives count
        LD      (NOMEN),A
        LD      HL,T845C
        SET     0,(HL)
        LD      HL,SCREEN
        LD      DE,SCREEN+1
        LD      BC,17FFh        ;Clear screen image
        LD      (HL),0
        LDIR
        LD      HL,0A000h       ;Title screen bitmap
        LD      DE,SCREEN
        LD      BC,4096
        LDIR
        LD      HL,SCREEN + 800h + 1*32 + 29
        LD      DE,MANDAT+64
        LD      C,0
        CALL    DRWFIX
        LD      HL,0FC00h       ;Attributes for the last room
        LD      DE,ATTR         ;(top third)
        LD      BC,256
        LDIR
        LD      HL,09E00h       ;Attributes for title screen
        LD      BC,512          ;(bottom two-thirds)
        LDIR
        LD      BC,31
        DI
        XOR     A
R8621:
        IN      E,(C)
        OR      E
        DJNZ    R8621   ;$-03
        AND     20h
        JR      NZ,R862F ;$+07
        LD      A,1
        LD      (KEMP),A
R862F:
        LD      IY,T846E
        CALL    C92DC
        JP      NZ,L8684
        XOR     A
        LD      (EUGHGT),A
```

C++ COMMENTS

Commenting in C++ involves using a double forward slash '/', or a forward slash and an asterisk, '/*'. You've already seen some brief examples but this is how they work.

STEP 1 Using the Hello World code as an example, you can easily comment on different sections of the code using the double forward slash:

```
//My first C++ program
cout << "Hello, world!\n";
```

```
#include <iostream>
using namespace std;

int main()
{
    //My first C++ program
    cout << "Hello, world!\n";


}
```

STEP 2 However, you can also add comments to the end of a line of code, to describe in a better way what's going on:

```
cout << "Hello, world!\n"; //This line outputs the words 'Hello, world!'. The \n denotes a new line.
```

Note, you don't have to put a semicolon at the end of a comment. This is because it's a line in the code that's ignored by the compiler.

```
#include <iostream>
using namespace std;

int main()
{
    //My first C++ program
    cout << "Hello, world!\n"; //This line outputs the words 'Hello, world!'. The
}
```

STEP 3 You can comment out several lines by using the forward slash and asterisk:

```
/* This comment can
     cover several lines
     without the need to add more slashes */
```

Just remember to finish the block comment with the opposite asterisk and forward slash.

```
int main()
{
    //My first C++ program
    cout << "Hello, world!\n";
    cout << "And greetings from C++";

    /* This comment can
       cover several lines
       without the need to add more slashes */


}
```

STEP 4 Be careful when commenting, especially with block comments. It's very easy to forget to add the closing asterisk and forward slash and thus negate any code that falls inside the comment block.

```
#include <iostream>
using namespace std;

int main()
{
    //My first C++ program
    cout << "Hello, world!\n";
    cout << "And greetings from C++";

    /* This comment can
       cover several lines
       without the need to add more slashes

    cout << "This line is now being ignored by the compiler!";

}
```

STEP 5 Obviously if you try and build and execute the code it errors out, complaining of a missing curly bracket '}' to finish off the block of code. If you've made the error a few times, then it can be time consuming to go back and rectify. Thankfully, the colour coding in Code::Blocks helps identify comments from code.

```
#include <iostream>
using namespace std;

int main()
{
    //My first C++ program
    cout << "Hello, world!\n";
    cout << "And greetings from C++";

    /* This comment can
       cover several lines
       without the need to add more slashes

    cout << "This line is now being ignored by the compiler!";

}
```

STEP 6 If you're using block comments, it's good practise in C++ to add an asterisk to each new line of the comment block. This also helps you to remember to close the comment block off before continuing with the code:

```
/* This comment can
 *    cover several lines
 *    without the need to add more slashes */
```

```
int main()
{
    //My first C++ program
    cout << "Hello, world!\n";
    cout << "And greetings from C++\n";

   /* This comment can
    *    cover several lines
    *    without the need to add more slashes */


    cout << "This line is now being ignored by the compiler!\n";

}
```

Variables

Variables differ slightly when using C++ as opposed to Python. In Python, you can simply state that 'a' equals 10 and a variable is assigned. However, in C++ a variable has to be declared with its type before it can be used.

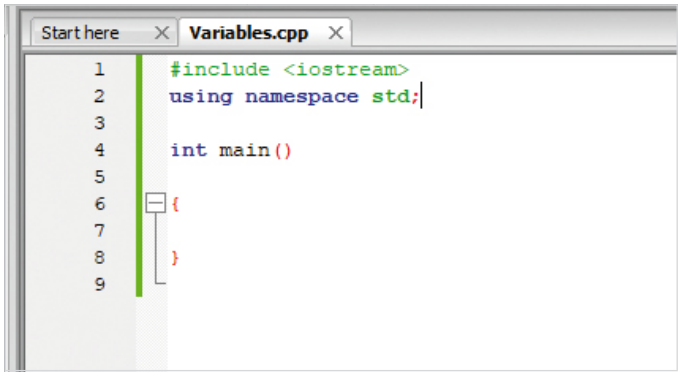
THE DECLARATION OF VARIABLES

You can declare a C++ variable by using statements within the code. There are several distinct types of variables you can declare. Here's how it works.

STEP 1

Open up a new, blank C++ file and enter the usual code headers:

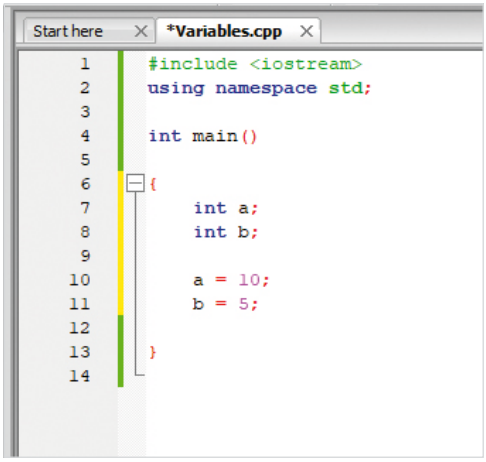
```
#include <iostream>
using namespace std;

int main()
{

}
```



STEP 2

Start simple by creating two variables, a and b, with one having a value of 10 and the other 5. You can use the data type int to declare these variables. Within the curly brackets, enter:

```
int a;
int b;

a = 10;
b = 5;
```

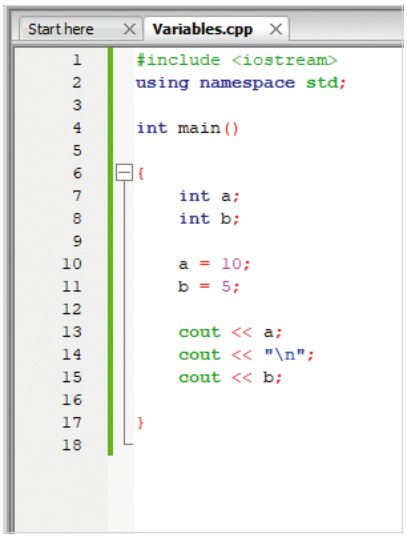


STEP 3

You can build and run the code but it won't do much, other than store the values 10 and 5 to the integers a and b. To output the contents of the variables, add:

```
cout << a;
cout << "\n";
cout << b;
```

The cout << "\n"; part simply places a new line between the output of 10 and 5.

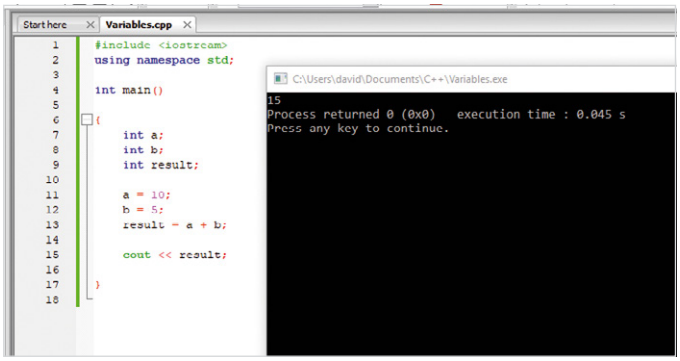


STEP 4

Naturally you can declare a new variable, call it result and output some simple arithmetic:

```
int result;

result = a + b;
cout << result;
```

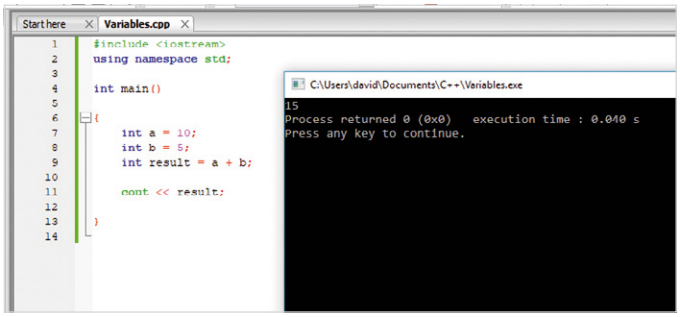
Insert the above into the code as per the screenshot.



STEP 5

You can assign a value to a variable as soon as you declare it. The code you've typed in could look like this, instead:

```
int a = 10;
int b = 5;
int result = a + b;

cout << result;
```



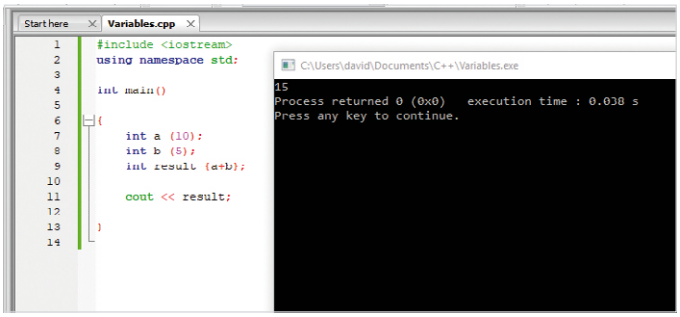
STEP 6

Specific to C++, you can also use the following to assign values to a variable as soon as you declare them:

```
int a (10);
int b (5);
```

Then, from the C++ 2011 standard, using curly brackets:

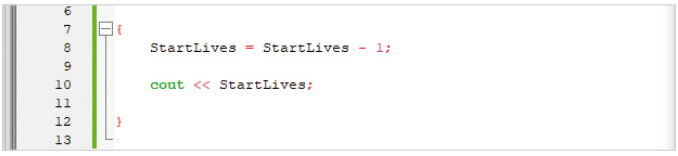
```
int result {a+b};
```



STEP 7

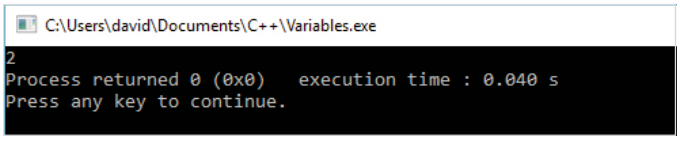
You can create global variables, which are variables that are declared outside any function and used in any function within the entire code. What you've used so far are local variables: variables used inside the function. For example:

```
#include <iostream>
using namespace std;
int StartLives = 3;

int main ()

{
      startLives = StartLives – 1;
      cout << StartLives;
}
```



STEP 8

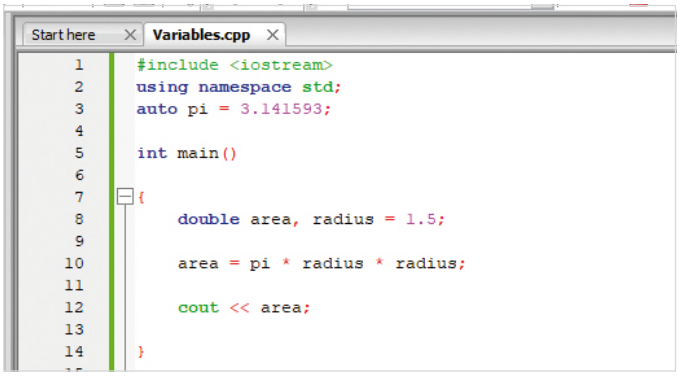
The previous step creates the variable StartLives, which is a global variable. In a game, for example, a player's lives go up or down depending on how well or how bad they're doing. When the player restarts the game, the StartLives returns to its default state: 3. Here we've assigned 3 lives, then subtracted 1, leaving 2 lives left.



STEP 9

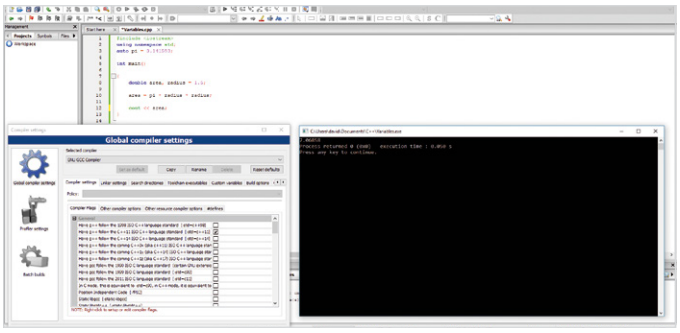
The modern C++ compiler is far more intelligent than most programmers give it credit. While there are numerous data types you can declare for variables, you can in fact use the auto feature:

```
#include <iostream>
using namespace std;
auto pi = 3.141593;

int main()

{
    double area, radius = 1.5;

    area = pi * radius * radius;

    cout << area;

}
```



STEP 10

A couple of new elements here: first, auto won't work unless you go to Settings > Compiler and tick the box labelled 'Have G++ follow the C++11 ISO C++ Language Standard [-std=c++1]'. Then, the new data type, double, which means double-precision floating point value. Enable C++11, then build and run the code. The result should be 7.06858.



Data Types

Variables, as we've seen, store information that the programmer can then later call up, and manipulate if required. Variables are simply reserved memory locations that store the values the programmer assigns, depending on the data type used.

THE VALUE OF DATA

There are many different data types available for the programmer in C++, such as an integer, floating point, Boolean, character and so on. It's widely accepted that there are seven basic data types, often called Primitive Built-in Types; however, you can create your own data types should the need ever arise within your code.

The seven basic data types are:

TYPE	COMMAND
Integer	`Integer`
Floating Point	`float`
Character	`char`
Boolean	`bool`
Double Floating Point	`double`
Wide Character	`wchar_t`
No Value	`void`

These basic types can also be extended using the following modifiers: Long, Short, Signed and Unsigned. Basically this means the modifiers can expand the minimum and maximum range values for each data type. For example, the int data type has a default value range of -2147483648 to 2147483647, a fair value, you would agree.

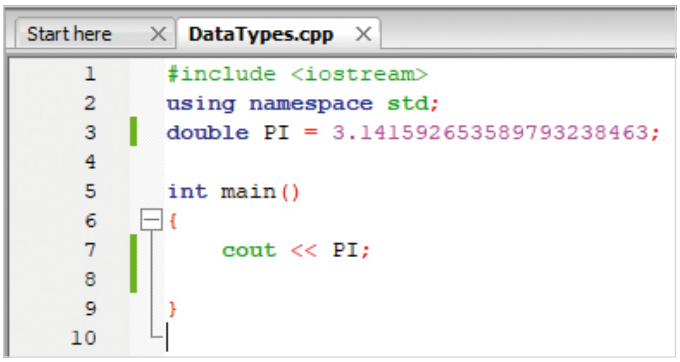
Now, if you were to use one of the modifiers, the range alters:

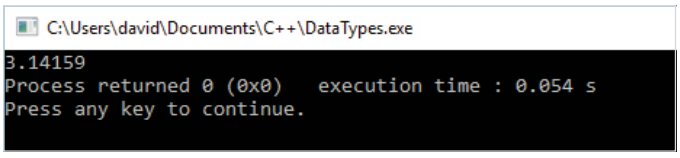
Unsigned int = 0 to 4294967295
Signed int = -2147483648 to 2147483647
Short int = -32768 to 32767
Unsigned Short int = 0 to 65,535
Signed Short int = -32768 to 32767
Long int = -2147483647 to 2147483647
Signed Long int = -2147483647 to 2147483647
Unsigned Long int = 0 to 4294967295

Naturally you can get away with using the basic type without the modifier, as there's plenty of range provided with each data type. However, it's considered good C++ programming practise to use the modifiers when possible.

There are issues when using the modifiers though. Double represents a double-floating point value, which you can use for incredibly accurate numbers but those numbers are only accurate up to the fifteenth decimal place. There's also the problem when displaying such numbers in C++ using the cout function, in that cout by default only outputs the first five decimal places. You can combat that by adding a cout.precision () function and adding a value inside the brackets, but even then you're still limited by the accuracy of the double data type. For example, try this code:

```
#include <iostream>
using namespace std;
double PI = 3.141592653589793238463;

int main()
{
    cout << PI;
}
```



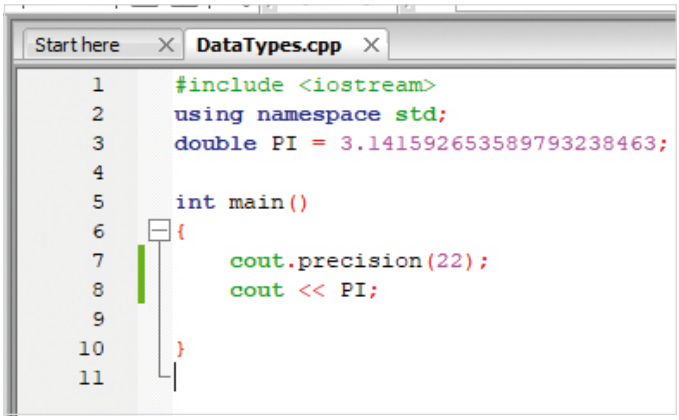


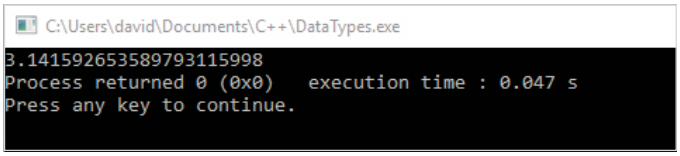
Build and run the code and as you can see the output is only 3.14159, representing cout's limitations in this example.

You can alter the code including the aforementioned cout.precision function, for greater accuracy. Take precision all the way up to 22 decimal places, with the following code:

```
#include <iostream>
using namespace std;
double PI = 3.141592653589793238463;

int main()
{
```

```
    cout.precision(22);
    cout << PI;
}
```



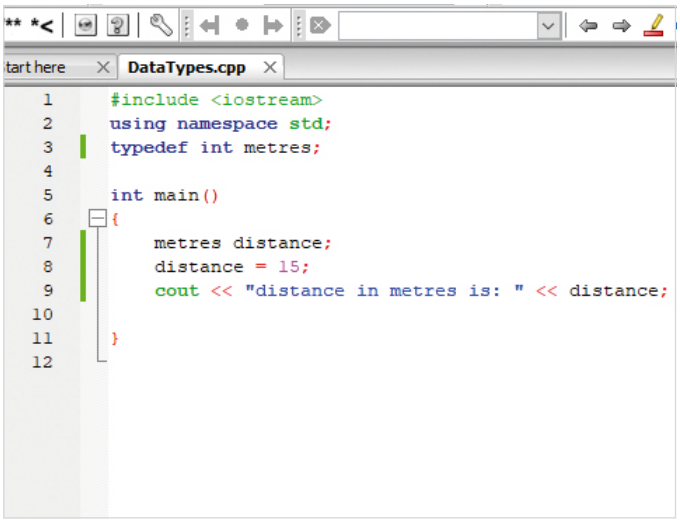


Again, build and run the code; as you can see from the command line window, the number represented by the variable PI is different to the number you've told C++ to use in the variable. The output reads the value of PI as 3.1415926535897931159 8, with the numbers going awry from the fifteenth decimal place.

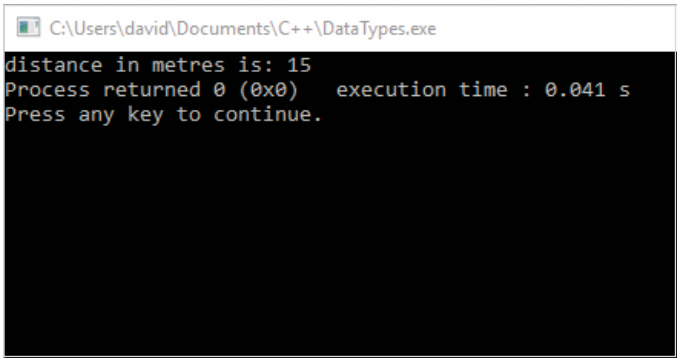
This is mainly due to the conversion from binary in the compiler and that the IEEE 754 double precision standard occupies 64-bits of data, of which 52-bits are dedicated to the significant (the significant digits in a floating-point number) and roughly 3.5-bits are taken holding the values 0 to 9. If you divide 53 by 3.5, then you arrive at 15.142857 recurring, which is 15-digits of precision.

To be honest, if you're creating code that needs to be accurate to more than fifteen decimal places, then you wouldn't be using C++, you would use some scientific specific language with C++ as the connective tissue between the two languages.

You can create your own data types, using an alias-like system called typedef. For example:




```
#include <iostream>
using namespace std;
typedef int metres;

int main()
{
    metres distance;
    distance = 15;
    cout << “distance in metres is: “ << distance;

}
```



This code when executed creates a new int data type called metres. Then, in the main code block, there's a new variable called distance, which is an integer; so you're basically telling the compiler that there's another name for int. We assigned the value 15 to distance and displayed the output: distance in metres is 15.

It might sound a little confusing to begin with but the more you use C++ and create your own code, the easier it becomes.

Strings

Strings are objects that represent and hold sequences of characters. For example, you could have a universal greeting in your code 'Welcome' and assign that as a string to be called up wherever you like in the program.

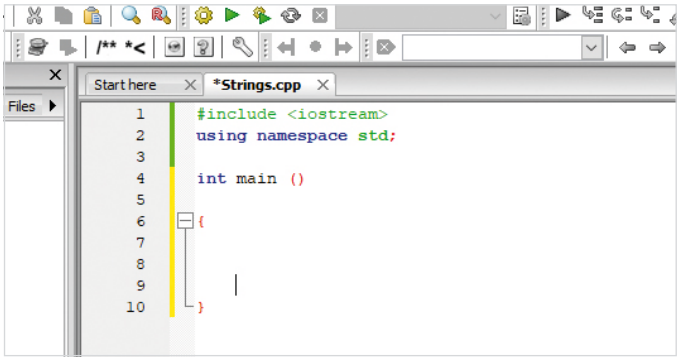
STRING THEORY

There are different ways in which you can create a string of characters, which historically are all carried over from the original C language, and are still supported by C++.

STEP 1

To create a string you use the char function. Open a new C++ file and begin with the usual header:

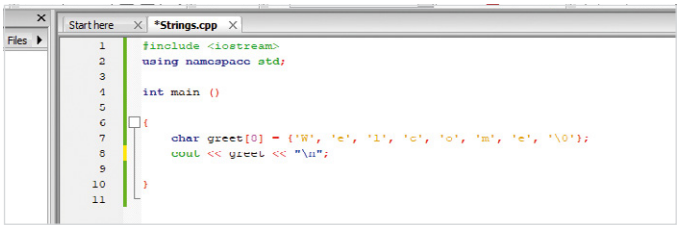
```
#include <iostream>
using namespace std;

int main ()

{

}
```



STEP 2

It's easy to confuse a string with an array. Here's an array, which can be terminated with a null character:

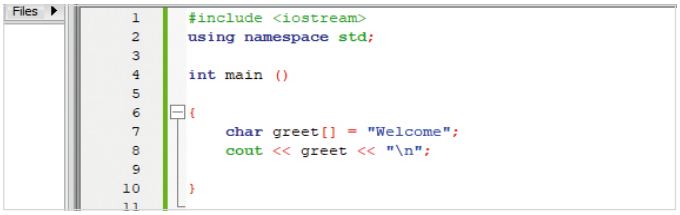
```
#include <iostream>
using namespace std;

int main ()

{
    char greet[8] = {'W', 'e', 'l', 'c', 'o', 'm', 'e', '\0'};
    cout << greet << "\n";

}
```



STEP 3

Build and run the code, and 'Welcome' appears on the screen. While this is perfectly fine, it's not a string. A string is a class, which defines objects that can be represented as a stream of characters and doesn't need to be terminated like an array. The code can therefore be represented as:

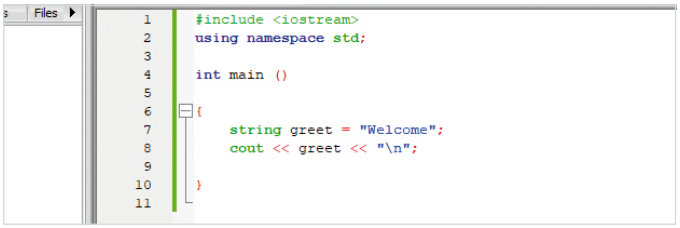
```
#include <iostream>
using namespace std;

int main ()

{
    char greet[] = "Welcome";
    cout << greet << "\n";

}
```



STEP 4

In C++ there's also a string function, which works in much the same way. Using the greeting code again, you can enter:

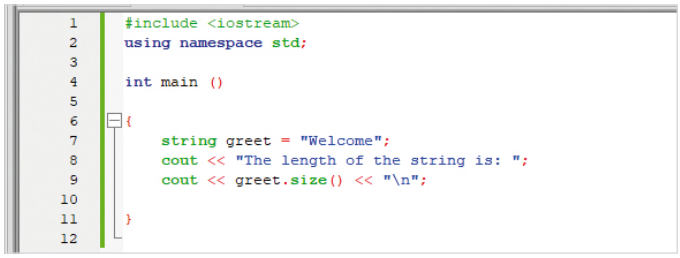
```
#include <iostream>
using namespace std;

int main ()

{
    string greet = "Welcome";
    cout << greet << "\n";

}
```



STEP 5

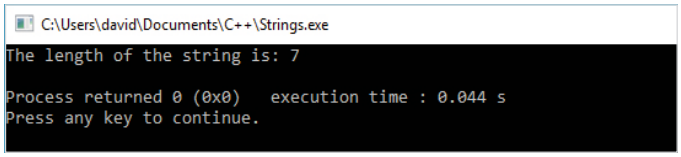
There are also many different operations that you can apply with the string function. For instance, to get the length of a string you can use:

```
#include <iostream>
using namespace std;

int main ()

{
    string greet = “Welcome”;
    cout << “The length of the string is: “;
    cout << greet.size() << “\n”;

}
```



STEP 6

You can see that we used greet.size() to output the length, the number of characters there are, of the contents of the string. Naturally, if you call your string something other than greet, then you need to change the command to reflect this. It’s always stringname.operation. Build and run the code to see the results.



STEP 7

You can of course add strings together, or rather combine them to form longer strings:

```
#include <iostream>
using namespace std;

int main ()

{
    string greet1 = “Hello”;
    string greet2 = “, world!”;
    string greet3 = greet1 + greet2;

    cout << greet3 << “\n”;

}
```

STEP 8

Just as you might expect, you can mix in an integer and store something to do with the string. In this example, we created int length, which stores the result of string. size() and outputs it to the user:

```
#include <iostream>
using namespace std;

int main ()

{
    int length;
    string greet1 = “Hello”;
    string greet2 = “, world!”;
    string greet3 = greet1 + greet2;

    length = greet3.size();

    cout << “The length of the combined strings
is: “ << length << “\n”;

}
```

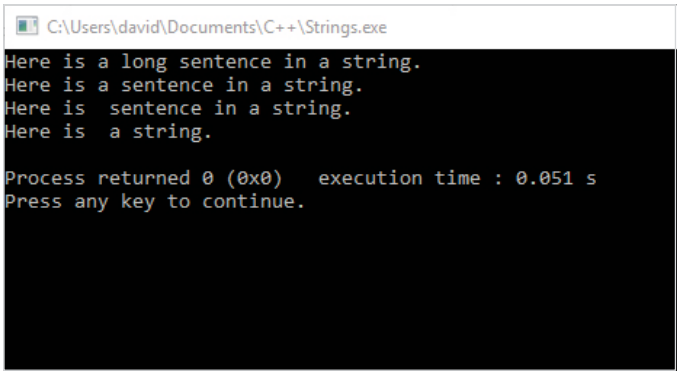
STEP 9

Using the available operations that come with the string function, you can manipulate the contents of a string. For example, to remove characters from a string you could use:

```
#include <iostream>
using namespace std;

int main ()

{
  string strg (“Here is a long sentence in a
string.”);
  cout << strg << ‘\n’;

  strg.erase (10,5);
  cout << strg << ‘\n’;

  strg.erase (strg.begin()+8);
  cout << strg << ‘\n’;

  strg.erase (strg.begin()+9, strg.end()-9);
  cout << strg << ‘\n’;

}
```

STEP 10

It’s worth spending some time playing around with the numbers, which are the character positions in the string. Occasionally, it can be hit and miss whether you get it right, so practice makes perfect. Take a look at the screenshot to see the result of the code.



C++ Maths

Programming is mathematical in nature and as you might expect, there's plenty of built-in scope for some quite intense maths. C++ has a lot to offer someone who's implementing mathematical models into their code. It can be extremely complex or relatively simple.

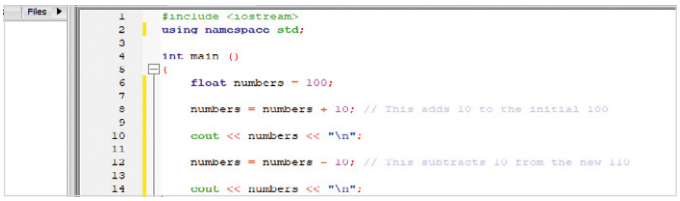
C++ = MC²

The basic mathematical symbols apply in C++ as they do in most other programming languages. However, by using the C++ Math Library, you can also calculate square roots, powers, trig and more.

STEP 1

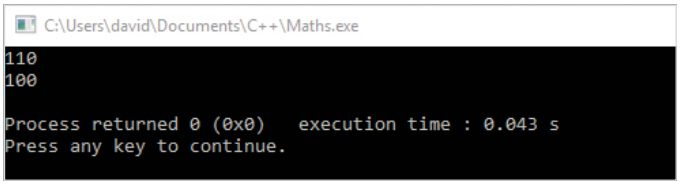
C++'s mathematical operations follow the same patterns as those taught in school, in that multiplication and division take precedence over addition and subtraction. You can alter that though. For now, create a new file and enter:

```
#include <iostream>
using namespace std;

int main ()
{
    float numbers = 100;

    numbers = numbers + 10; // This adds 10 to the initial 100

    cout << numbers << "\n";

    numbers = numbers - 10; // This subtracts 10 from the new 110

    cout << numbers << "\n";

}
```



STEP 2

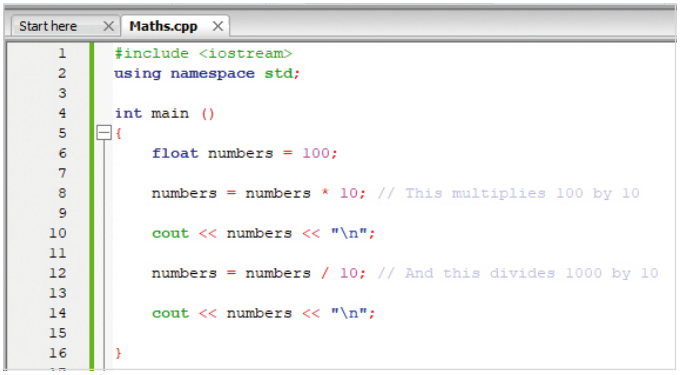
While simple, it does get the old maths muscle warmed up. Note that we used a float for the numbers variable. While you can happily use an integer, if you suddenly started to use decimals, you would need to change to a float or a double, depending on the accuracy needed. Run the code and see the results.



STEP 3

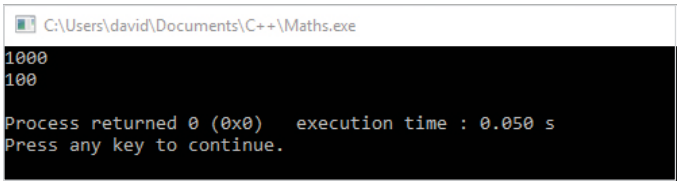
Multiplication and division can be applied as such:

```
#include <iostream>
using namespace std;

int main ()
{
    float numbers = 100;

    numbers = numbers * 10; // This multiplies 100 by 10

    cout << numbers << "\n";

    numbers = numbers / 10; // And this divides 1000 by 10

    cout << numbers << "\n";

}
```



STEP 4

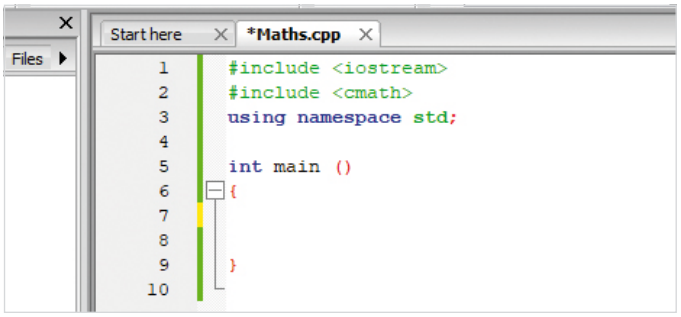
Again, execute the simple code and see the results. While not particularly interesting, it's a start into C++ maths. We used a float here, so you can play around with the code and multiply by decimal places, as well as divide, add and subtract.



STEP 5

The interesting maths content comes when you call upon the C++ Math Library. Within this header are dozens of mathematical functions along with further operations. Everything from computing cosine to arc tangent with two parameters, to the value of PI. You can call the header with:

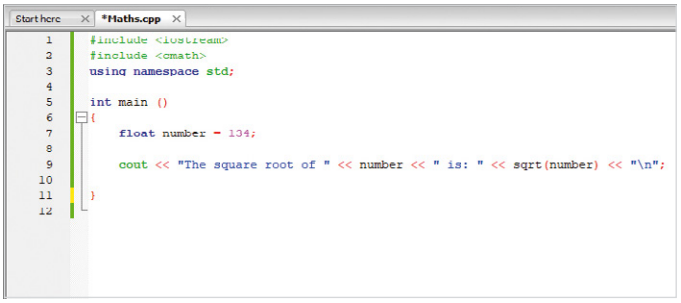
```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{

}
```



STEP 6

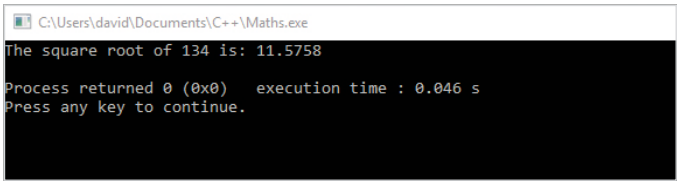
Start by getting the square root of a number:

```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{
    float number = 134;

    cout << “The square root of “ << number << “ is: “ << sqrt(number) << “\n”;

}
```



STEP 7

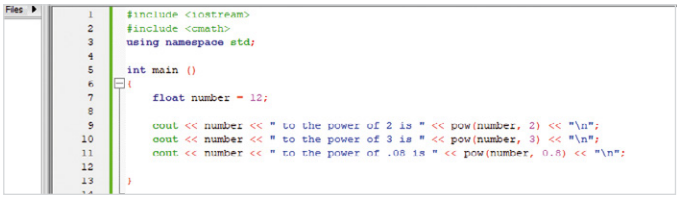
Here we created a new float called number and used the sqrt(number) function to display the square root of 134, the value of the variable, number. Build and run the code, and your answer reads 11.5758.



STEP 8

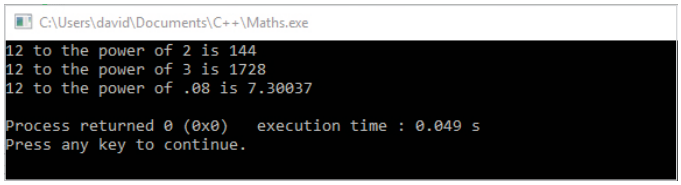
Calculating powers of numbers can be done with:

```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{
    float number = 12;

    cout << number << “ to the power of 2 is “ << pow(number, 2) << “\n”;
    cout << number << “ to the power of 3 is “ << pow(number, 3) << “\n”;
    cout << number << “ to the power of .08 is “ << pow(number, 0.8) << “\n”;

}
```



STEP 9

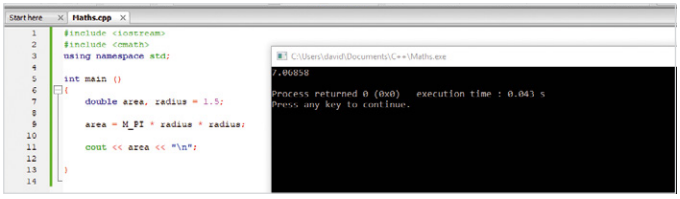
Here we created a float called number with the value of 12, and the pow(variable, power) is where the calculation happens. Of course, you can calculate powers and square roots without using variables. For example, pow (12, 2) outputs the same value as the first cout line in the code.



STEP 10

The value of Pi is also stored in the cmath header library. It can be called up with the M_PI function. Enter cout << M_PI; into the code and you get 3.14159; or you can use it to calculate:

```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{
    double area, radius = 1.5;

    area = M_PI * radius * radius;

    cout << area << “\n”;

}
```



"Measuring programming progress by lines of code is like measuring aircraft building progress by weight."

– Bill Gates
(Co-founder of Microsoft)

C++ Input/ Output

There's a satisfying feeling when you program code that asks the user for input, then uses that input to produce something that the user can see. Even if it's simply asking for someone's name and displaying a personal welcome message, it's a big leap forward and creates a more human level of interaction with your code.

User interaction, character literals and defining constants, alongside file input and output, are all covered in the following pages. All of which will help you to better understand how a C++ program works.

User Interaction

There's nothing quite as satisfying as creating a program that responds to you. This basic user interaction is one of the most taught aspects of any language and with it you're able to do much more than simply greet the user by name.

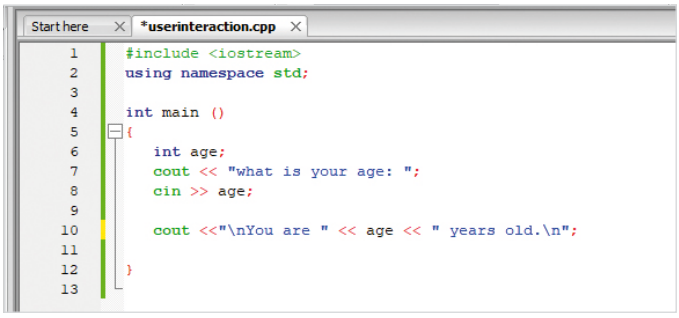
HELLO, DAVE

You have already used cout, the standard output stream, throughout our code. Now you're going to be using cin, the standard input stream, to prompt a user response.

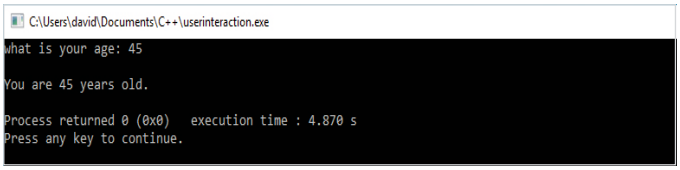
STEP 1 Anything that you want the user to input into the program needs to be stored somewhere in the system memory, so it can be retrieved and used. Therefore, any input must first be declared as a variable, so it's ready to be used by the user. Start by creating a blank C++ file with headers.

STEP 2 The data type of the variable must also match the type of input you want from the user. For example, to ask a user their age, you would use an integer like this:

```
#include <iostream>
using namespace std;

int main ()
{
   int age;
   cout << "what is your age: ";
   cin >> age;

   cout <<"\nYou are " << age << " years old.\n";

}
```

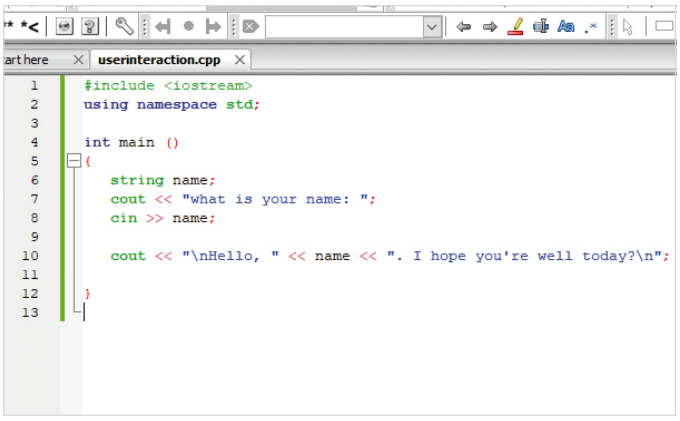


STEP 3 The cin command works in the opposite way from the cout command. With the first cout line you're outputting 'What is your age' to the screen, as indicated with the chevrons. Cin uses opposite facing chevrons, indicating an input. The input is put into the integer age and called up in the second cout command. Build and run the code.



STEP 4 If you're asking a question, you need to store the input as a string; to ask the user their name, you would use:

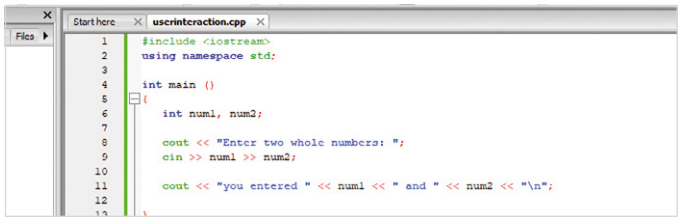
```
#include <iostream>
using namespace std;

int main ()
{
   string name;
   cout << "what is your name: ";
   cin >> name;

   cout << "\nHello, " << name << ". I hope you're well today?\n";

}
```



STEP 5 The principal works the same as the previous code. The user's input, their name, is stored in a string, because it contains multiple characters, and retrieved in the second cout line. As long as the variable 'name' doesn't change, then you can recall it wherever you like in your code.

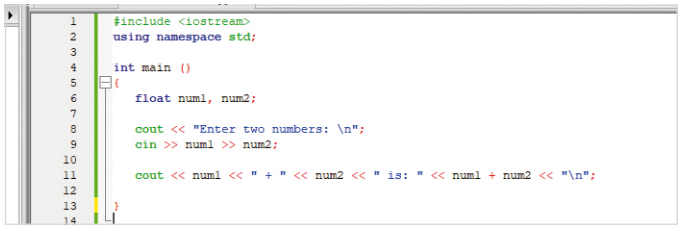
STEP 6 You can chain input requests to the user but just make sure you have a valid variable to store the input to begin with. Let's assume you want the user to enter two whole numbers:

```
#include <iostream>
using namespace std;

int main ()
{
   int num1, num2;

   cout << "Enter two whole numbers: ";
   cin >> num1 >> num2;

   cout << "you entered " << num1 << " and " << num2 << "\n";

}
```



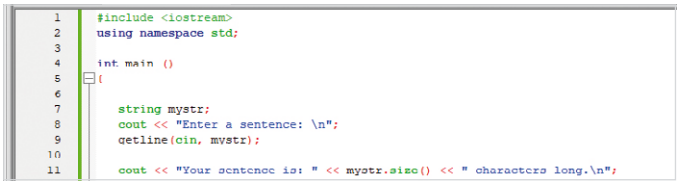
STEP 7 Likewise, inputted data can be manipulated once you have it stored in a variable. For instance, ask the user for two numbers and do some maths on them:

```
#include <iostream>
using namespace std;

int main ()
{
   float num1, num2;

   cout << "Enter two numbers: \n";
   cin >> num1 >> num2;

   cout << num1 << " + " << num2 << " is: " << num1 + num2 << "\n";

}
```

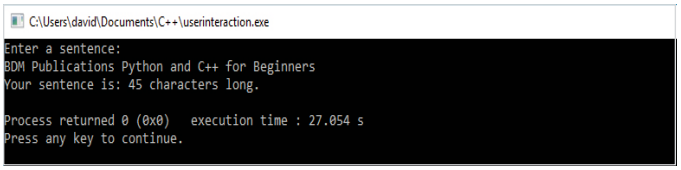


STEP 8 While cin works well for most input tasks, it does have a limitation. Cin always considers spaces as a terminator, so it's designed for just single words not multiple words. However, getline takes cin as the first argument and the variable as the second:

```
#include <iostream>
using namespace std;

int main ()
{
   string mystr;
   cout << "Enter a sentence: \n";
   getline(cin, mystr);

   cout << "Your sentence is: " << mystr.size() << " characters long.\n";

}
```

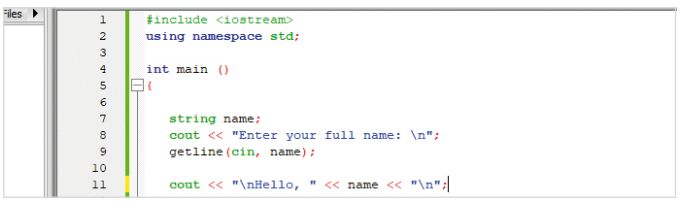


STEP 9 Build and execute the code, then enter a sentence with spaces. When you're done the code reads the number of characters. If you remove the getline line and replace it with cin >> mystr and try again, the result displays the number of characters up to the first space.



STEP 10 Getline is usually a command that new C++ programmers forget to include. The terminating white space is annoying when you can't figure out why your code isn't working. In short, it's best to use getline(cin, variable) in future:

```
#include <iostream>
using namespace std;

int main ()
{
   string name;
   cout << "Enter your full name: \n";
   getline(cin, name);

   cout << "\nHello, " << name << "\n";

}
```



Character Literals

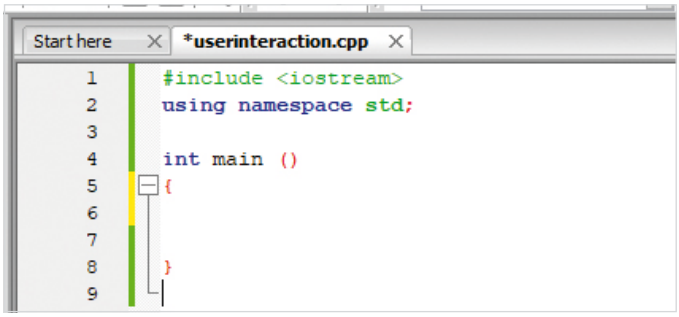
In C++ a literal is an object or variable that once defined remains the same throughout the code. However, a character literal is defined by a backslash, such as the \n you've been using at the end of a cout statement to signify a new line.

ESCAPE SEQUENCE

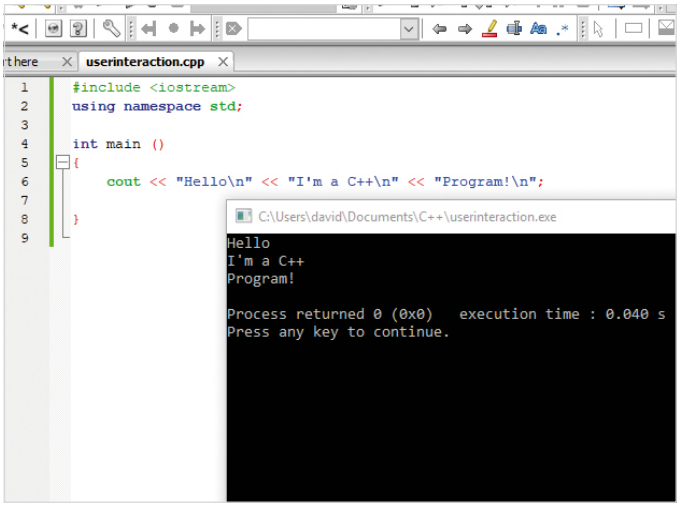
When used in something like a cout statement, character literals are also called escape sequence codes. They allow you to insert a quote, an alert, new line and much more.

STEP 1 Create a new C++ file and enter the relevant headers:

```
#include <iostream>
using namespace std;

int main ()
{

}
```

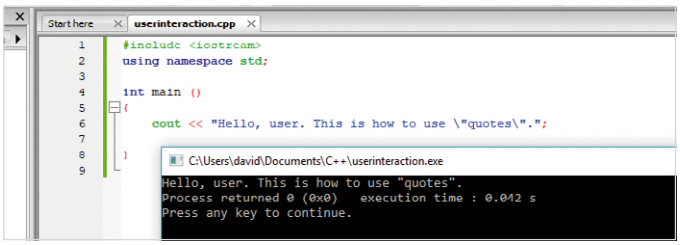


STEP 2 You've already experienced the \n character literal placing a new line wherever it's called. The line: **cout << "Hello\n" << "I'm a C++\n" << "Program\n";** outputs three lines of text, each starting after the last \n.



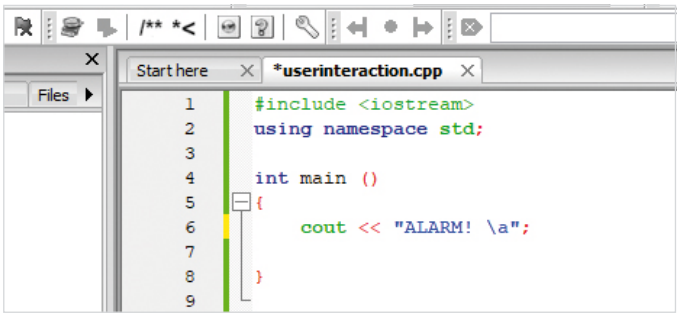
STEP 3 If you wanted to insert speech quotes inside a cout statement, you would have to use a backslash as it already uses quotes:

```
#include <iostream>
using namespace std;

int main ()
{
    cout << "Hello, user. This is how to use
\"quotes\".";

}
```



STEP 4 There's even a character literal that can trigger an alarm. In Windows 10, it's the notification sound that chimes when you use \a. Try this code, and turn up your sound.

```
#include <iostream>
using namespace std;

int main ()
{
    cout << "ALARM! \a";

}
```



A HANDY CHART

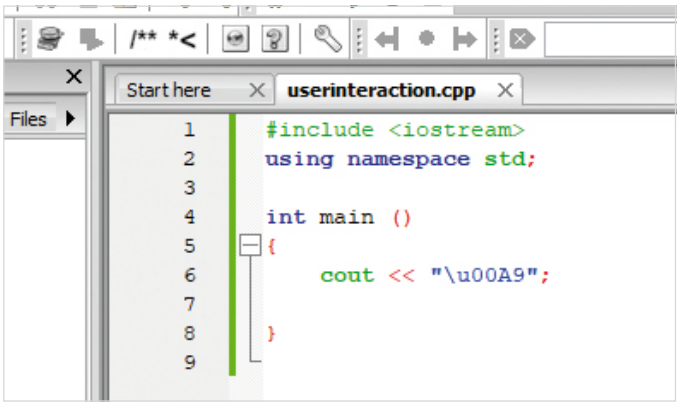
There are numerous character literals, or escape sequence codes, to choose from. We therefore thought it would be good for you to have a handy chart available, for those times when you need to insert a code.

ESCAPE SEQUENCE CODE	CHARACTER
\\	Backslash
\'	Single Quote
\"	Double Quote (Speech Marks)
\?	Question Mark
\a	Alert/Alarm
\b	Backspace
\f	Form Feed
\n	New Line
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab
\0	Null Character
\uxxxx	Unicode (UTF-8)
\Uxxxxxxxx	Unicode (UTF-16)

UNICODE CHARACTERS (UTF-8)

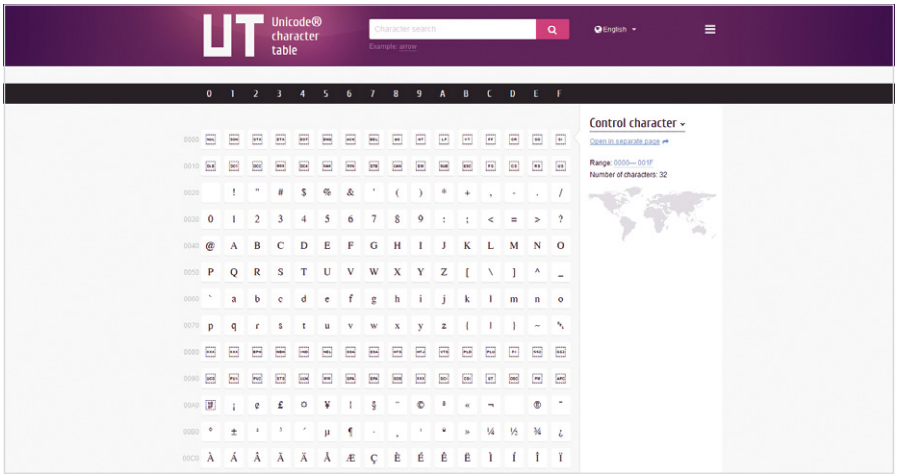
Unicode characters are symbols or characters that are standard across all platforms. For example, the copyright symbol, that can be entered via the keyboard by entering the Unicode code, followed by ALT+X. In the case of the copyright symbol enter: 00A9 Alt+X. In C++ code, you would enter:

```
#include <iostream>
using namespace std;

int main ()
{
    cout << "\u00A9";
```



UNICODE CHARACTER TABLE

A complete list of the available Unicode characters can be found at **www.unicode-table.com/en/**. Hover your mouse over the character to see the unique code to enter in C++. While it may be a little overwhelming to look at to begin with, bookmark the page as you will probably need to come back to it for reference as you dig deeper into C++, and indeed character literals. One more thing, the table will also display characters from different languages, such as Tibetan or Sudanese. This means your code can be truly universal.



Defining Constants

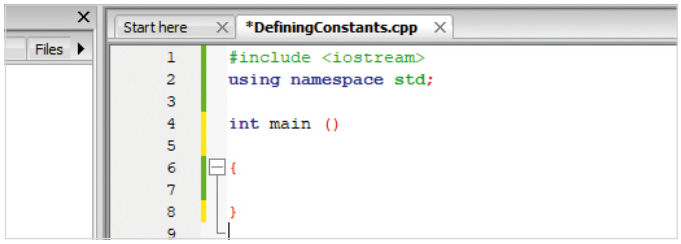
Constants are fixed values in your code. They can be any basic data type but as the name suggests their value remains constant throughout the entire code. There are two separate ways to define a constant in C++, the #define pre-processor and const.

#DEFINE

The pre-processors are instructions to the compiler to pre-process the information before it goes ahead and compiles the code. #include is a pre-processor as is #define.

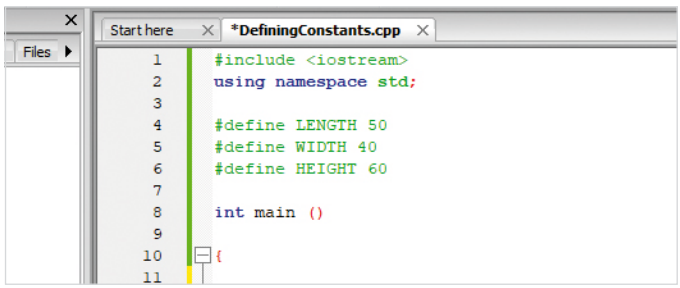
STEP 1 You can use the #define pre-processor to define any constants you want in our code. Start by creating a new C++ file complete with the usual headers:

```
#include <iostream>
using namespace std;

int main ()

{

}
```



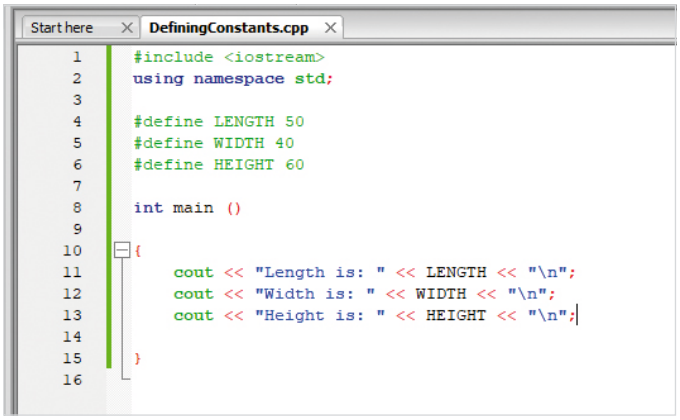
STEP 2 Now let's assume your code has three different constants: length, width and height. You can define them with:

```
#include <iostream>
using namespace std;
#define LENGTH 50
#define WIDTH 40
#define HEIGHT 60

int main ()

{

}
```

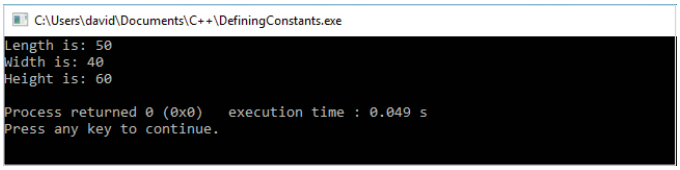


STEP 3 Note the capitals for defined constants, it's considered good programming practise to define all constants in capitals. Here, the assigned values are 50, 40 and 60, so let's call them up:

```
#include <iostream>
using namespace std;

#define LENGTH 50
#define WIDTH 40
#define HEIGHT 60

int main ()

{
    cout << "Length is: " << LENGTH << "\n";
    cout << "Width is: " << WIDTH << "\n";
    cout << "Height is: " << HEIGHT << "\n";
}
```



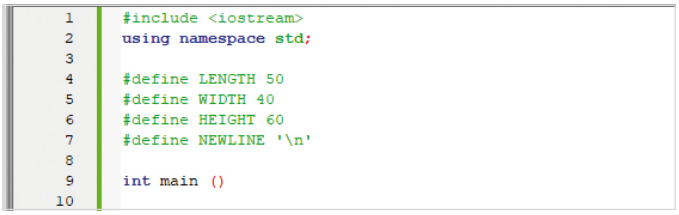
STEP 4 Build and run the code. Just as expected, it displays the values for each of the constants created. It's worth noting that you don't need a semicolon when you're defining a constant with the #define keyword.



STEP 5

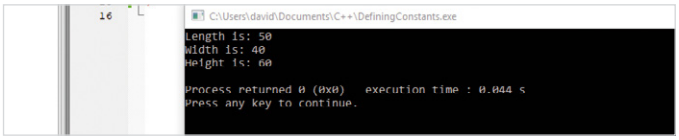
You can also define other elements as a constant. For example, instead of using \n for a newline in the cout statement, you can define it at the start of the code:

```
#include <iostream>
using namespace std;

#define LENGTH 50
#define WIDTH 40
#define HEIGHT 60
#define NEWLINE ‘\n’

int main ()

{
    cout << “Length is: “ << LENGTH << NEWLINE;
    cout << “Width is: “ << WIDTH << NEWLINE;
    cout << “Height is: “ << HEIGHT << NEWLINE;
}
```



STEP 6

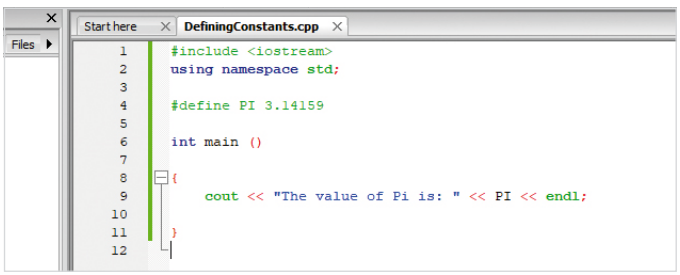
The code, when built and executed, does exactly the same as before, using the new constant NEWLINE to insert a newline in the cout statement. Incidentally, creating a newline constant isn’t a good idea unless you’re making it smaller than \n or even the endl command.



STEP 7

Defining a constant is a good way of initialising your base values at the start of your code. You can define that your game has three lives, or even the value of PI without having to call up the C++ math library:

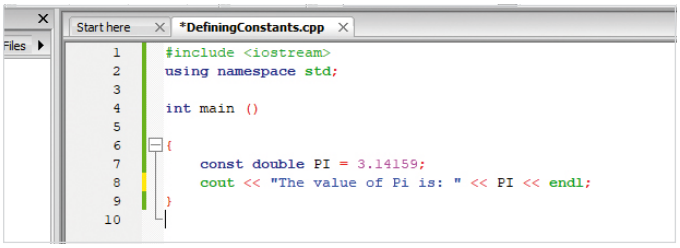
```
#include <iostream>
using namespace std;

#define PI 3.14159

int main ()

{
    cout << “The value of Pi is: “ << PI << endl;

}
```



STEP 8

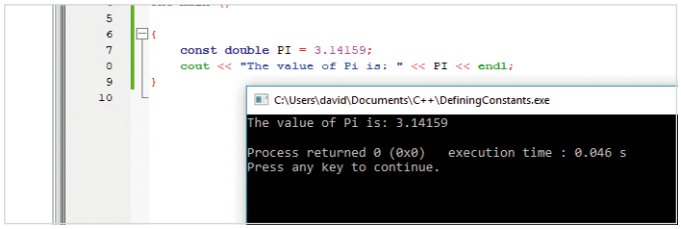
Another method of defining a constant is with the const keyword. Use const together with a data type, variable and value: const type variable = value. Using Pi as an example:

```
#include <iostream>
using namespace std;

int main ()

{
    const double PI = 3.14159;
    cout << “The value of Pi is: “ << PI << endl;
}
```



STEP 9

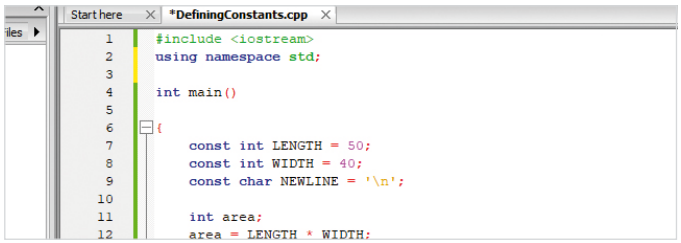
Because you’re using const within the main block of code, you need to finish the line with a semicolon. You can use either, as long as the names and values don’t clash, but it’s worth mentioning that #define requires no memory, so if you’re coding to a set amount of memory, #define is your best bet.



STEP 10

Const works in much the same way as #define. You can create static integers and even newlines:

```
#include <iostream>
using namespace std;

int main()

{
    const int LENGTH = 50;
    const int WIDTH = 40;
    const char NEWLINE = ‘\n’;

    int area;
    area = LENGTH * WIDTH;

    cout << “Area is: “ << area << NEWLINE;

}
```



File Input/Output

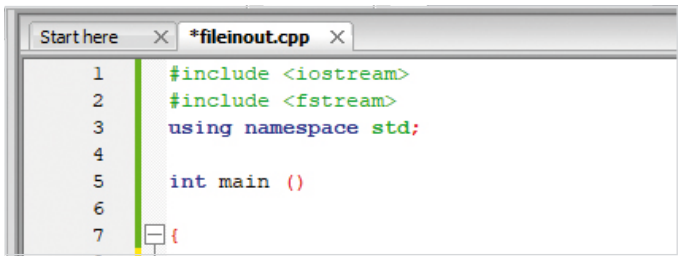
The standard iostream library provides C++ coders with the cin and cout input and output functionality. However, to be able to read and write from a file you need to utilise another C++ library, called fstream.

FSTREAMS

There are two main data types within the fstream library that are used to open a file, read from it and write to it, ofstream and ifstream. Here's how they work.

STEP 1 The first task is to create a new C++ file and along with the usual headers you need to include the new fstream header:

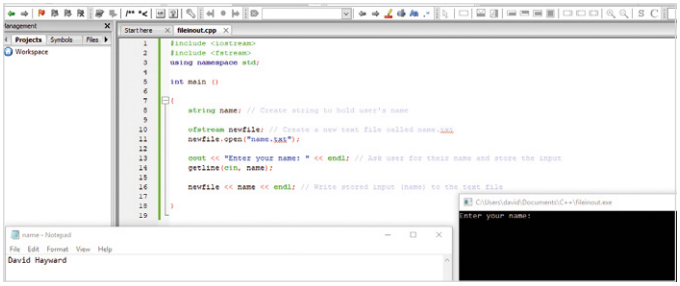
```
#include <iostream>
#include <fstream>
Using namespace std;

int main ()

{

}
```



STEP 2 Begin by asking a user for their name and writing that information to a file. You need the usual string to store the name, and getline to accept the input from the user.

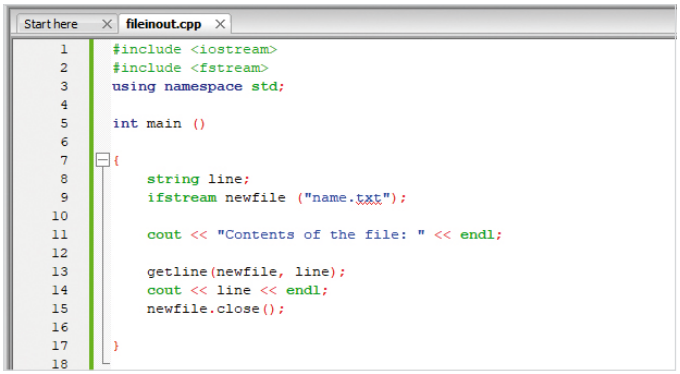
```
#include <iostream>
#include <fstream>
using namespace std;

int main ()

{

    string name;

    ofstream newfile;
    newfile.open("name.txt");

    cout << "Enter your name: " << endl;
    getline(cin, name);

    newfile << name << endl;

    newfile.close();

}
```

STEP 3 We've included comments in the screenshot of step 2 to help you understand the process. You created a string called name, to store the user's inputted name. You also created a text file called name.txt (with the ofstream newfile and newfile.open lines), asked the user for their name and stored it and then written the data to the file.



STEP 4 To read the contents of a file, and output it to the screen, you need to do things slightly differently. First you need to create a string variable to store the file's contents (line by line), then open the file, use getline to read the file line by line and output those lines to the screen. Finally, close the file.

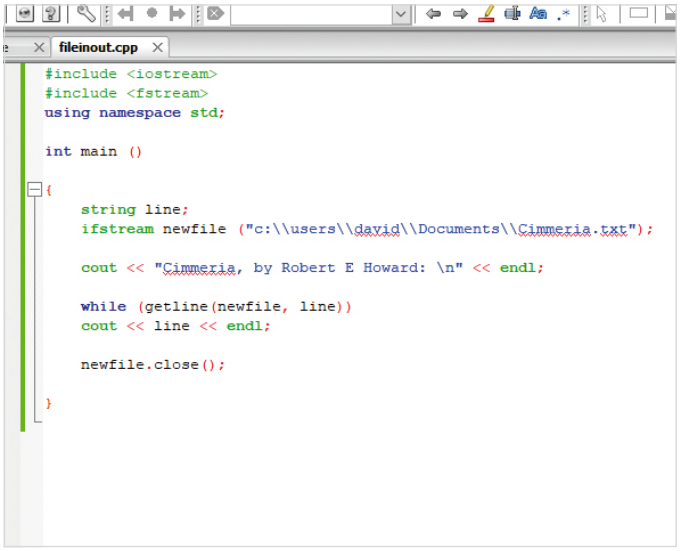
```
    string line;
    ifstream newfile ("name.txt");

    cout << "Contents of the file: " << endl;

    getline(newfile, line);
    cout << line << endl;
    newfile.close();
```



STEP 5

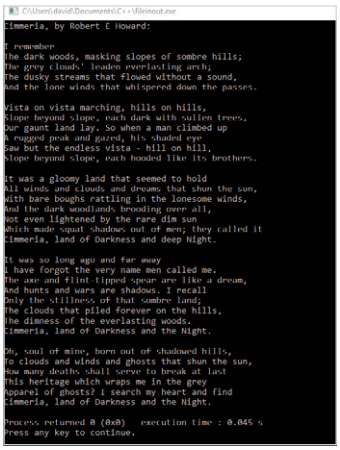
The code above is great for opening a file with one or two lines but what if there are multiple lines? Here we opened a text file of the poem Cimmeria, by Robert E Howard:

```
    string line;
    ifstream newfile ("c:\\users\\david\\
Documents\\Cimmeria.txt");

    cout << "Cimmeria, by Robert E Howard: \n" <<
endl;

    while (getline(newfile, line))
    cout << line << endl;

    newfile.close();
```



STEP 6

You can no doubt see that we've included a while loop, which we cover in a few pages time. It means that while there are lines to be read from the text file, C++ getlines them. Once all the lines are read, the output is displayed on the screen and the file is closed.



STEP 7

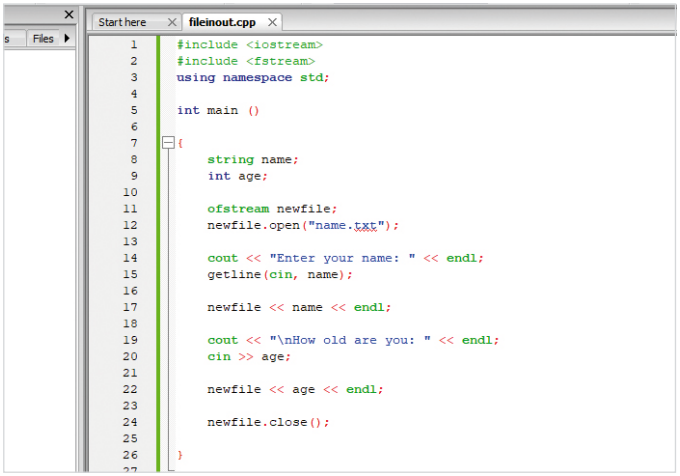
You can also see that the location of the text file Cimmeria.txt isn't in the same folder as the C++ program. When we created the first name.txt file, it was written to the same folder where the code was located; this is done by default. To specify another folder, you need to use double-back slashes, as per the character literals/escape sequence code.

```
6
7   {
8       string line;
9       ifstream newfile ("c:\\users\\david\\Documents\\Cimmeria.txt");
10
11      cout << "Cimmeria, by Robert E Howard: \n" << endl;
```

STEP 8

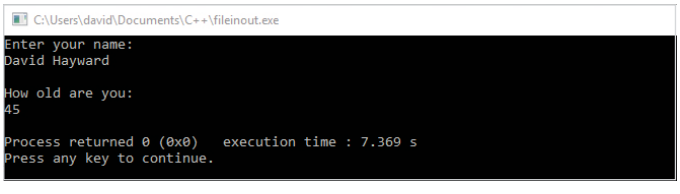
Just as you might expect, you can write almost anything you like to a file, for reading either in Notepad or via the console through the C++ code:

```
    string name;
    int age;

    ofstream newfile;
    newfile.open("name.txt");

    cout << "Enter your name: " << endl;
    getline(cin, name);

    newfile << name << endl;

    cout << "\nHow old are you: " << endl;
    cin >> age;

    newfile << age << endl;

    newfile.close();
```



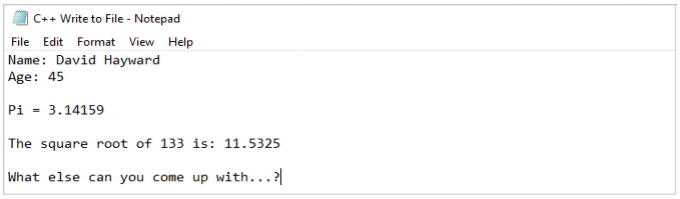
STEP 9

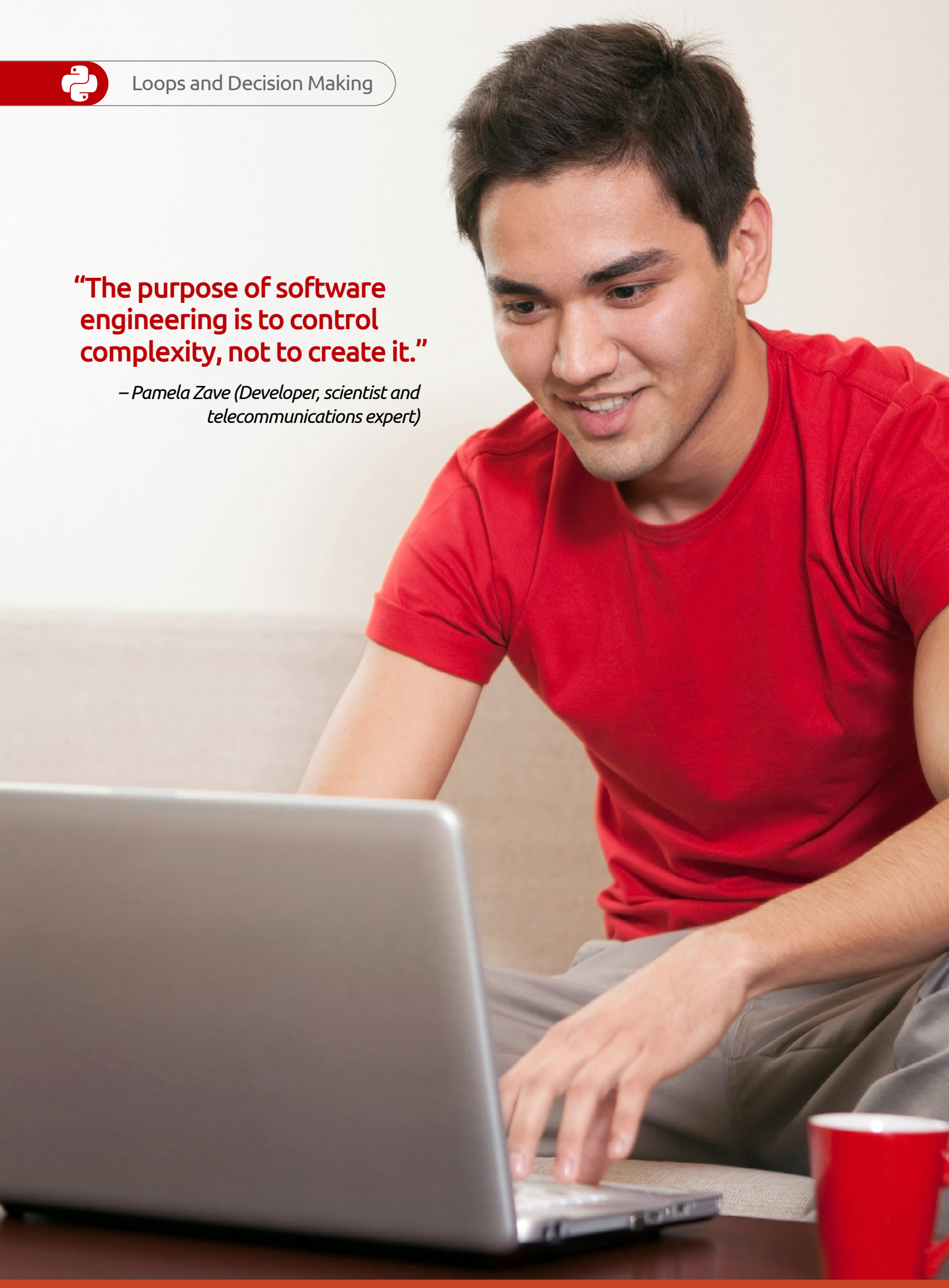
The code from step 8 differs again, but only where it comes to adding the age integer. Notice that we used cin >> age, instead of the previous getline(cin, variable). The reason for this is that the getline function handles strings, not integers; so when you're using a data type other than a string, use the standard cin.



STEP 10

Here's an exercise: see if you can create code to write several different elements to a text file. You can have a user's name, age, phone number etc. Maybe even the value of Pi and various mathematical elements. It's all good practice.





"The purpose of software engineering is to control complexity, not to create it."

– Pamela Zave (Developer, scientist and telecommunications expert)

Loops and Decision Making

Loops and repetition are one of the most important factors of any programming language. Good use of a loop will create a program that does exactly what you want it to and delivers the desired outcome without issues or errors.

Without loops and decision-making events within the code, your program will never be able to offer the user any choice. It's this understanding of choice that elevates your skills as a programmer and makes for much better code.

While Loop

A while loop's function is to repeat a statement, or a group of statements, while a certain condition remains true. When the while loop starts, it initialises itself by testing the condition of the loop and the statements within, before executing the rest of the loop.

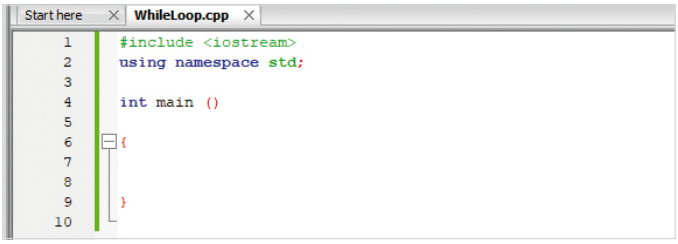
TRUE OR FALSE?

While loops are one of the most popular form of C++ code looping. They repeatedly run the code contained within the loop while the condition is true. Once it proves false, the code continues as normal.

STEP 1

Clear what you've done so far and create a new C++ file. There's no need for any extra headers at the moment, so add the standard headers as per usual:

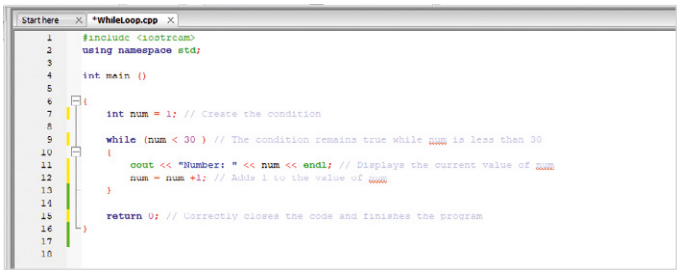
```
#include <iostream>
using namespace std;

int main ()

{

}
```



STEP 2

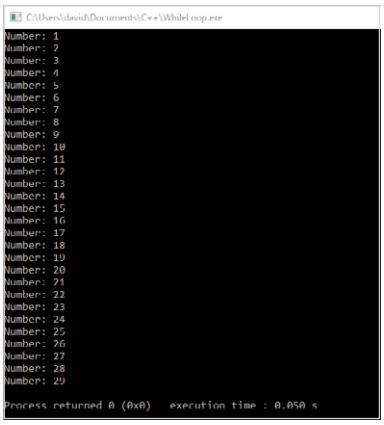
Create a simple while loop. Enter the code below, build and run (we've added comments to the screen shot):

```
{
    int num = 1;

    while (num < 30 )
    {
        cout << "Number: " << num << endl;
        num = num +1;
    }

    return 0;
}
```



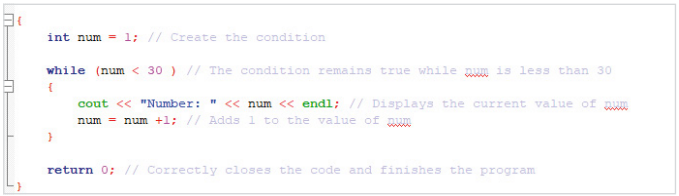
STEP 3

First you need to create a condition, so use a variable called num and give it the value 1. Now create the while loop, stating that as long as num is less than 30, the loop is true. Within the loop the value of num is displayed and adds 1 until it's more than 30.



STEP 4

We're introducing a few new elements here. The first are the opening and closing braces for the while loop. This is because our loop is a compound statement, meaning a group of statements; note also, there's no semicolon after the while statement. You now also have return 0, which is a clean and preferred way of ending the code.



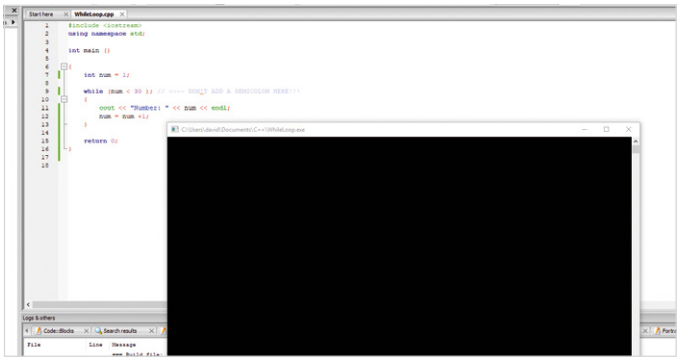
STEP 5

If you didn't need to see the continually increasing value of num, you could have done away with the compound while statement and instead just added num by itself until it reached 30, and then displayed the value:

```
{
    int num = 1;

    while (num < 30 )
        num++;
    cout << "Number: " << num << endl;

    return 0;
}
```

STEP 6 It's important to remember not to add a semicolon at the end of a while statement. Why? Well, as you know, the semicolon represents the end of a C++ line of code. If you place one at the end of a while statement, your loop will be permanently stuck until you close the program.

STEP 7 In our example, if we were to execute the code the value of num would be 1, as set by the int statement. When the code hits the while statement it reads that while the condition of 1 being less than 30 is true, loop. The semicolon closes the line, so the loop repeats; but it never adds 1 to num, as it won't continue through the compound statement.



STEP 8 You can manipulate the while statement to display different results depending on what code lies within the loop. For example, to read the poem, Cimmeria, word by word, you would enter:

```
#include <iostream>
#include <fstream>
using namespace std;

int main ()

{
    string word;
    ifstream newfile ("C:\\users\\david\\
Documents\\Cimmeria.txt");

    cout << "Cimmeria, by Robert E Howard: \n" <<
endl;

    while (newfile >> word)
    {
        cout << word << endl;
    }

    return 0;
}
```

STEP 9 You can further expand the code to enable each word of the poem to appear every second. To do so, you need to pull in a new library, <windows.h>. This is a Windows only library and within it you can use the Sleep() function:

```
#include <iostream>
#include <fstream>
#include <windows.h>
using namespace std;

int main ()

{
    string word;
    ifstream newfile ("C:\\users\\david\\
Documents\\Cimmeria.txt");

    cout << "Cimmeria, by Robert E Howard: \n" <<
endl;

    while (newfile >> word)
    {
        cout << word << endl;
        Sleep(1000);
    }

    return 0;
}
```

STEP 10 Sleep() works in milliseconds, so Sleep(1000) is one second, Sleep(10000) is ten seconds and so on. Combining the sleep function (along with the header it needs) and a while loop enables you to come up with some interesting countdown code.

```
#include <iostream>
#include <windows.h>
using namespace std;

int main ()

{
    int a = 10;

    while (a != 0)
    {
        cout << a << endl;
        a = a - 1;
        Sleep(1000);
    }

    cout << "\nBlast Off!" << endl;

    return 0;
}
```

For Loop

In some respects, a for loop works in a very similar way to that of a while loop, although it's structure is different. A for loop is split into three stages: an initialiser, a condition and an incremental step. Once set up, the loop repeats itself until the condition becomes false.

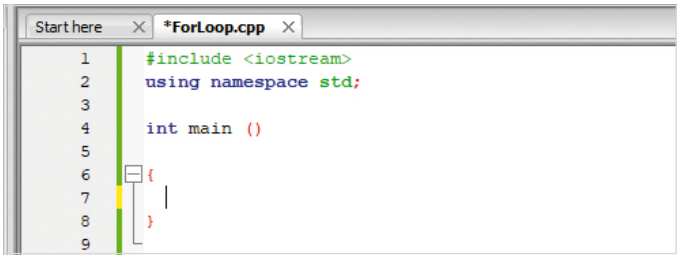
LOOPY LOOPS

The initialise stage of a for loop is executed only once and this sets the point reference for the loop. The condition is evaluated by the loop to see if it's true or false and then the increment is executed. The loop then repeats the second and third stage.

STEP 1

Create a new C++ file, with the standard headers:

```
#include <iostream>
using namespace std;

int main ()

{


}
```



STEP 2

Start simple and create a for loop that counts from 1 to 30, displaying the value to the screen with each increment:

```
{
    //For Loop Begins
    for( int num = 1; num < 30; num = num +1 )
    {
        cout << "Number: " << num << endl;
    }

    return 0;

}
```

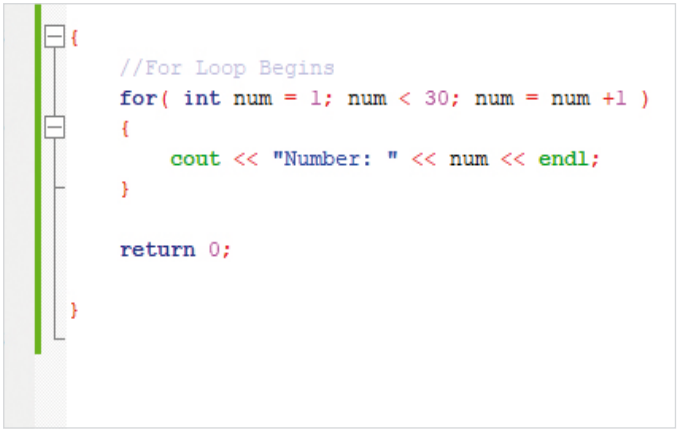
STEP 3

Working through the process of the for loop, begin by creating an integer called num and assigning it a value of 1. Next, set the condition, in this case num being less than 30. The last stage is where you create the increments; here it's the value of num being added by 1.

```
{
    //For Loop Begins
    for( int num = 1; num < 30; num = num +1 )
```

STEP 4

After the loop, you created a compound statement in braces (curly brackets), that displays the current value of the integer num. Every time the for loop repeats itself, the second and third stages of the loop, it adds 1 until the condition <30 is false. The loop then ends and the code continues, ending neatly with return 0.



```
{
    //For Loop Begins
    for( int num = 1; num < 30; num = num +1 )
    {
        cout << "Number: " << num << endl;
    }

    return 0;

}
```

STEP 5

A for loop is quite a neat package in C++, all contained within its own brackets, while the other elements outside of the loop are displayed below. If you want to create a 10-second countdown, you could use:

```
#include <iostream>
#include <windows.h>
using namespace std;

int main ()

{
    //For Loop Begins
    for( int a = 10; a != 0; a = a -1 )
    {
        cout << a << endl;
        Sleep(1000);
    }

    cout << "\nBlast Off!" << endl;

    return 0;

}
```

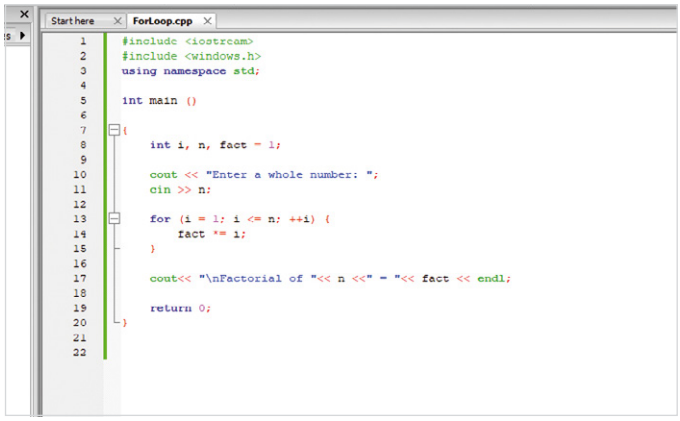
STEP 6

With the countdown code, don't forget to include the windows.h library, so you can use the Sleep command. Build and run the code; in the command console you can see the numbers 10 to 1 countdown in one second increments, until it reaches zero and Blast Off! appears.

STEP 7

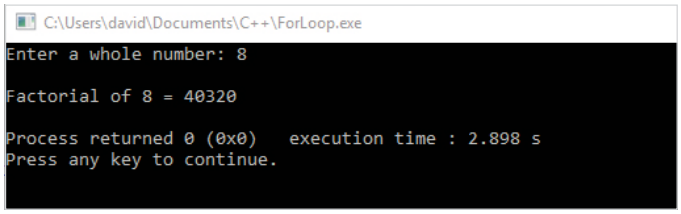
Naturally you can include a lot more content into a for loop, including some user input:

```
    int i, n, fact = 1;

    cout << "Enter a whole number: ";
    cin >> n;

    for (i = 1; i <= n; ++i) {
        fact *= i;
    }

    cout<< "\nFactorial of "<< n <<" = "<< fact << endl;

    return 0;
```



STEP 8

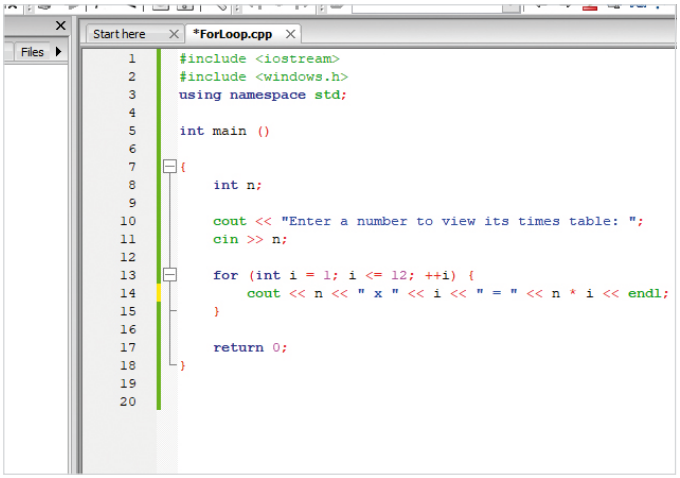
The code from step 7, when built and run, asks for a number, then displays the factorial of that number through the for loop. The user's number is stored in the integer n, followed by the integer I which is used to check if the condition is true or false, adding 1 each time and comparing it to the user's number, n.



STEP 9

Here's an example of a for loop displaying the multiplication tables of a user inputted number. Handy for students:

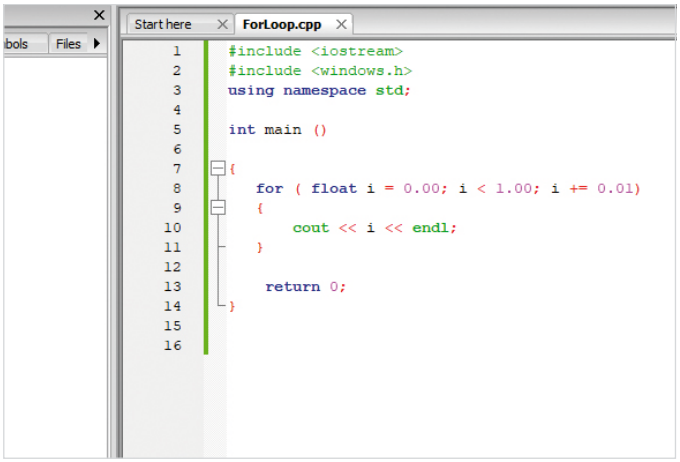
```
{
    int n;

    cout << "Enter a number to view its times table: ";
    cin >> n;

    for (int i = 1; i <= 12; ++i) {
        cout << n << " x " << i << " = " << n * i << endl;
    }

    return 0;
}
```



STEP 10

The value of the integer i can be expanded from 12 to whatever number you want, displaying a very large multiplication table in the process (or a small one). Of course the data type within a for loop doesn't have to be an integer; as long as it's valid, it works.

```
for ( float i = 0.00; i < 1.00; i += 0.01)
   {
       cout << i << endl;
   }

return 0;
```



Do… While Loop

A do… while loop differs slightly from that of a for or even a while loop. Both for and while set and examine the state of the condition at the start of the loop, or the top of the loop if you prefer. However, a do… while loop, is similar to a while loop but instead checks the condition at the bottom of the loop.

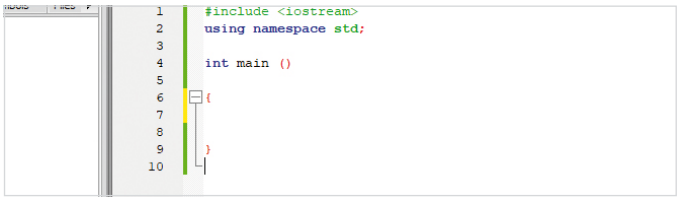
DO LOOPS

The good thing about a do… while loop is that it's guaranteed to run through at least once. It's structure is: do, followed by statements, while condition is true. This is how it works.

STEP 1

Begin with a new, blank C++ file and enter the standard headers:

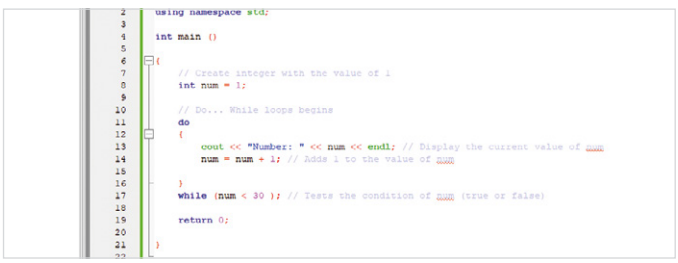
```
#include <iostream>
using namespace std;

int main ()

{


}
```



STEP 2

Begin with a simple number count:

```
{
    int num = 1;

    do
    {
        cout << "Number: " << num << endl;
        num = num + 1;

    }
    while (num < 30 );

    return 0;

}
```



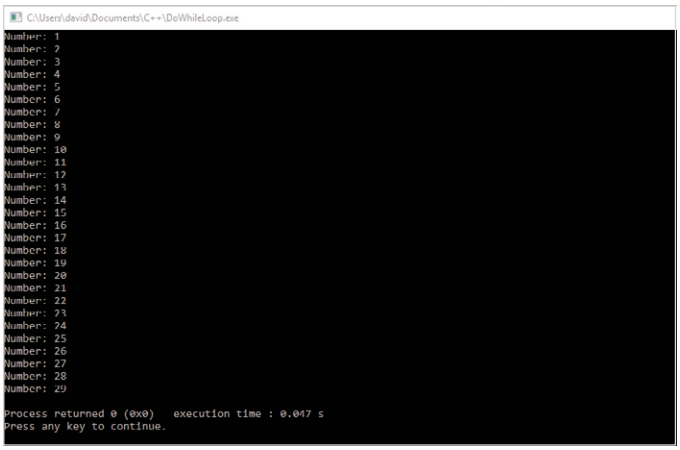
STEP 3

Now, here's a look at the structure of a do… while loop. First you create an integer called num, with the value of 1. Now the do… while loops begins. The code inside the body of the loop is executed at least once, then the condition is checked for either true or false.

```
// Create integer with the value of 1
int num = 1;

// Do... While loops begins
do
{
    cout << "Number: " << num << endl; // Display the current value of num
    num = num + 1; // Adds 1 to the value of num

}
while (num < 30 ); // Tests the condition of num (true or false)
```

STEP 4

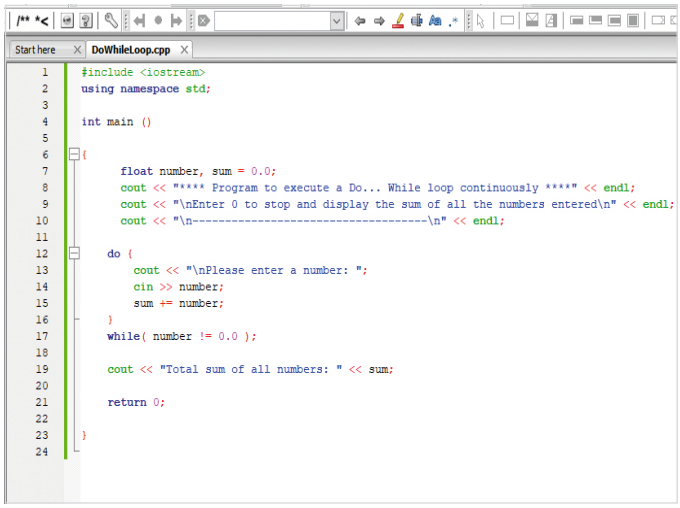
If the condition is true, the loop is executed. This continues until the condition is false. When the condition has been expressed as false, the loop terminates and the code continues. This means you can create a loop where the code continues until the user enters a certain character.



STEP 5

If you want code to add up user inputted numbers until the user enters zero:

```
{
      float number, sum = 0.0;
      cout << "**** Program to execute a Do...
While loop continuously ****" << endl;
      cout << "\nEnter 0 to stop and display the
sum of all the numbers entered\n" << endl;
      cout << "\n-----------------------------------
---\n" << endl;

    do {
        cout<<"\nPlease enter a number: ";
        cin>>number;
        sum += number;
    }
    while(number != 0.0);

    cout<<"Total sum of all numbers: "<<sum;

    return 0;
}
```



STEP 6

The code from Step 5 works as follows: two floating point variables are assigned, number and sum, both with the value of 0.0. There is a brief set of instructions for the user, then the do… while loop begins.

```
{
      float number, sum = 0.0;
      cout << "**** Program to execute a Do... While loop continuously ****" << endl;
      cout << "\nEnter 0 to stop and display the sum of all the numbers entered\n" << endl;
      cout << "\n-----------------------------------\n" << endl;
```

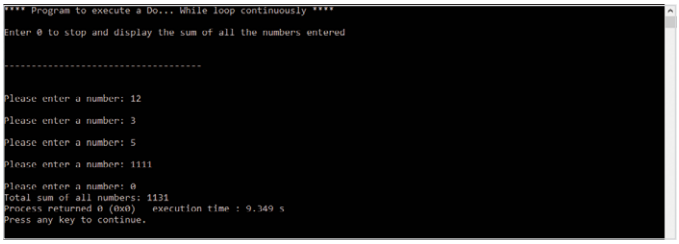
STEP 7

The do… while loop in this instance asks the user to input a number, which you assigned to the float variable, number. The calculation step uses the second floating point variable, sum, which adds the value of number every time the user enters a new value.

```
    do {
        cout << "\nPlease enter a number: ";
        cin >> number;
        sum += number;
```

STEP 8

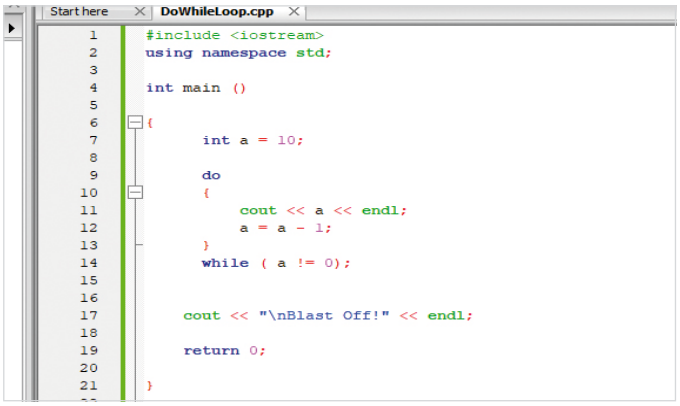
Finally, the while statement checks the condition of the variable number. If the user has entered zero, then the loop is terminated, if not then it continues indefinitely. When the user finally enters zero, the value of sum, the total value of all the user's input, is displayed. The loop, and the program, then ends.



STEP 9

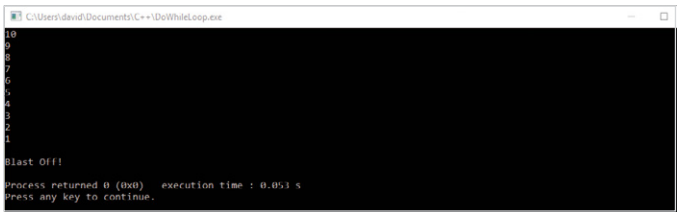
Using the countdown and Blast Off! code used previously, a do… while loop would look like:

```
{
      int a = 10;

      do
      {
          cout << a << endl;
          a = a - 1;
      }
      while ( a != 0);


    cout << "\nBlast Off!" << endl;

    return 0;

}
```



STEP 10

The main advantage of using a do… while loop is because it's an exit-condition loop; whereas a while loop is an entry-control loop. Therefore, if your code requires a loop that needs to be executed at least once (for example, to check the number of lives in a game), then a do… while loop is perfect.



If Statement

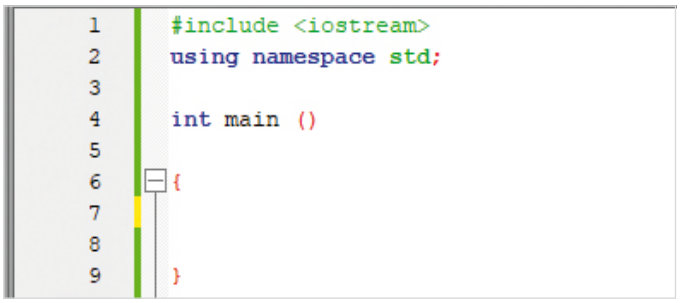
The decision making statement 'if' is probably one of the most used statements in any programming language, regardless of whether it's C++, Python, BASIC or anything else. It represents a junction in the code, where IF one condition is true, do this; or IF it's false, do that.

IF ONLY

If uses a Boolean expression within its statement. If the Boolean expression is true, the code within the statement is executed. If not, then the code after the statement is executed instead.

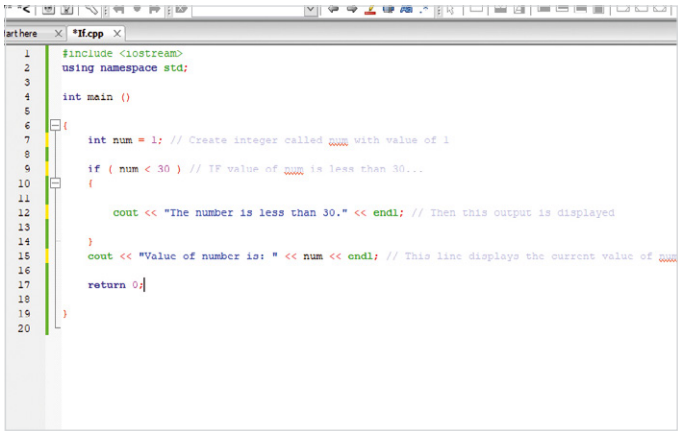
STEP 1 First, create a new C++ file and enter the relevant standard headers, as usual:

```
#include <iostream>
using namespace std;

int main ()

{


}
```

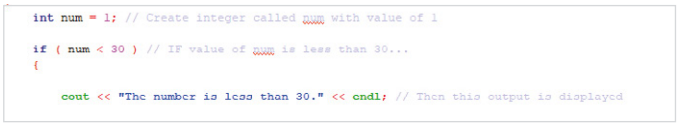


STEP 2 If is best explained when you use a number-based condition:

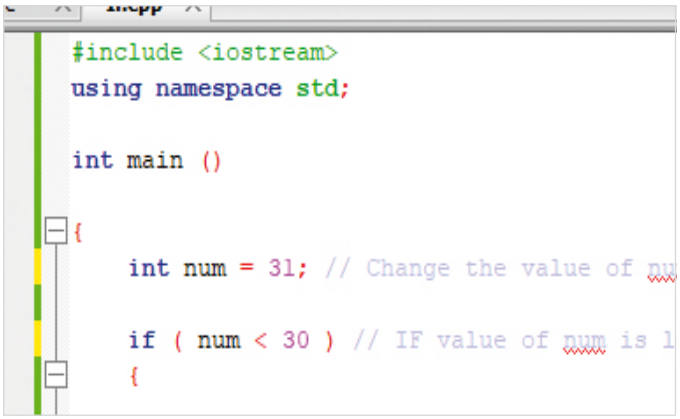
```
{
    int num = 1;

    if ( num < 30 )
    {

        cout << "The number is less than 30." <<
endl;

    }
    cout << "Value of number is: " << num << endl;

    return 0;

}
```



STEP 3 What's going on here? To begin, an integer called num was created and assigned with the value of 1. The if statement comes next, and in this case we've instructed the code that if the condition, the value, of num is less than 1, then the code within the braces should be executed.



STEP 4 The second cout statement displays the current value of num and the program terminates safely. It's easy to see how the if statement works if you were to change the initial value of num from 1 to 31.

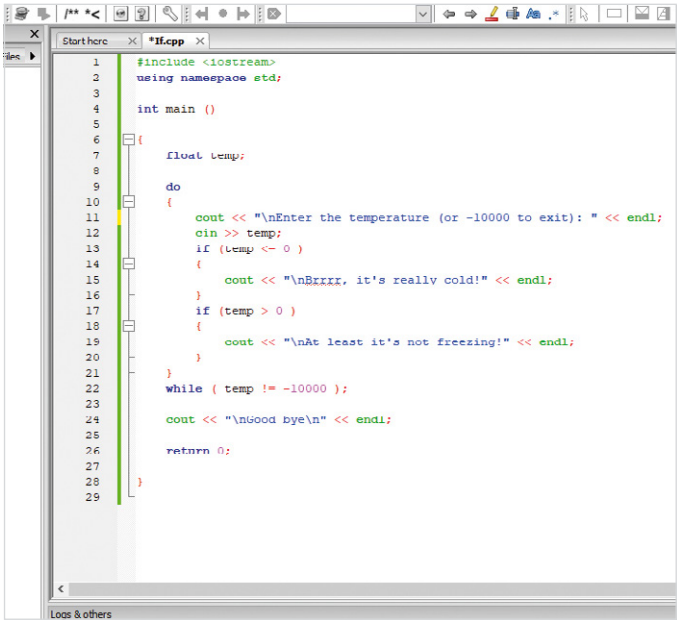


STEP 5 When you change the value to anything above 30, then build and run the code, you can see that the only line to be outputted to the screen is the second cout statement, displaying the current value of num. This is because the initial if statement is false, so it ignores the code within the braces.

```
Value of number is: 31

Process returned 0 (0x0)   execution time : 0.046 s
Press any key to continue.
```

STEP 6 You can include an if statement within a do… while loop. For example:

```
{
    float temp;

    do
    {
        cout << “\nEnter the temperature (or
-10000 to exit): “ << endl;
        cin >> temp;
        if (temp <= 0 )
        {
            cout << “\nBrrrr, it’s really cold!”
<< endl;
        }
        if (temp > 0 )
        {
            cout << “\nAt least it’s not
freezing!” << endl;
        }
    }
    while ( temp != -10000 );

    cout << “\nGood bye\n” << endl;

    return 0;

}
```



STEP 7 The code in Step 6 is simplistic but effective. First we created a floating point integer called temp, then a do… while loop that asks the user to enter the current temperature.

```
{
    float temp;

    do
    {
        cout << "\nEnter the temperature (or -10000 to exit): " << endl;
        cin >> temp;
```

STEP 8 The first if statement checks to see if the user's inputted value is less than or equal to zero. If it is, then the output is 'Brrrr, it's really cold!'. Otherwise, if the input is greater than zero, the code outputs 'At least it's not freezing!'.

```
    do
    {
        cout << "\nEnter the temperature (or -10000 to exit): " << endl;
        cin >> temp;
        if (temp <= 0 )
        {
            cout << "\nBrrrr, it's really cold!" << endl;
        }
        if (temp > 0 )
        {
            cout << "\nAt least it's not freezing!" << endl;
        }
```

STEP 9 Finally, if the user enters the value -10000, which is impossibly cold so is therefore a unrealistic value, the do… while loop is terminated and a friendly 'Good bye' is displayed to the screen.

```
{
    float temp;

    do
    {
        cout << "\nEnter the temperature (or -10000 to exit): " << endl;
        cin >> temp;
        if (temp <= 0 )
        {
            cout << "\nBrrrr, it's really cold!" << endl;
        }
        if (temp > 0 )
        {
            cout << "\nAt least it's not freezing!" << endl;
        }
    }
    while ( temp != -10000 );

    cout << "\nGood bye\n" << endl;

    return 0;

}
```

STEP 10 Using if is quite powerful, if it's used correctly. Just remember that if the condition is true then the code executes what's in the braces. If not, it continues on its merry way. See what else you can come up with using if and a combination of loops.

```
Enter the temperature (or -10000 to exit):
14

At least it's not freezing!

Enter the temperature (or -10000 to exit):
2

At least it's not freezing!

Enter the temperature (or -10000 to exit):
-110

Brrrr, it's really cold!

Enter the temperature (or -10000 to exit):
0

Brrrr, it's really cold!

Enter the temperature (or -10000 to exit):
-10000

Brrrr, it's really cold!

Good bye


Process returned 0 (0x0)   execution time : 17.323 s
Press any key to continue.
```

If… Else Statement

There is a much better way to use an if statement in your code, with if… else. If… else works in much the same way as a standard if statement. If the Boolean expression is true, the code within the braces is executed. Else, the code within the next set of braces is used instead.

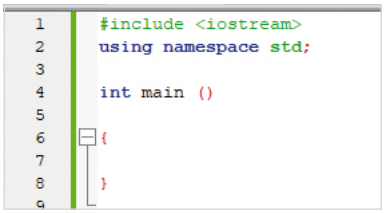
IF YES, ELSE NO

There are two sections of code that can be executed depending on the outcome in an if… else statement. It's quite easy to visualise once you get used to its structure.

STEP 1

Begin with a new C++ file and the standard headers:

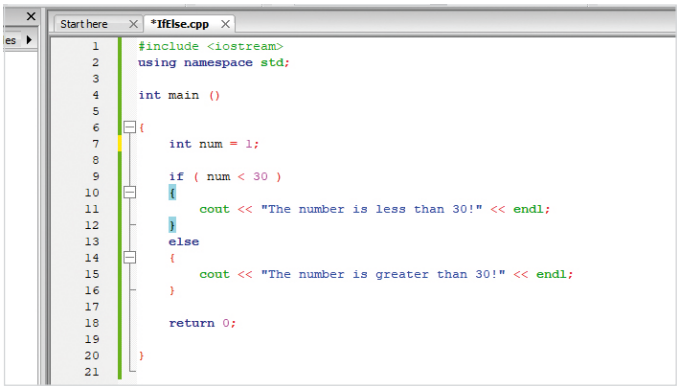
```
#include <iostream>
using namespace std;

int main ()

{

}
```



STEP 2

Let's expand the code from the If Statement on the previous page:

```
{
    int num = 1;

    if ( num < 30 )
    {
        cout << “The number is less than 30!” <<
endl;
    }
    else
    {
        cout << “The number is greater than 30!”
<< endl;
    }

    return 0;

}
```



STEP 3

The first line in the code creates the integer called num and gives it a value of 1. The if statement checks to see if the value of num is less than thirty and if it is it outputs “The number is less than 30!” to the console.

```
    if ( num < 30 )
    {
        cout << "The number is less than 30!" << endl;
    }
```

STEP 4

The else companion to if checks if the number is greater than 30 and if so, then displays “The number is greater than 30!” to the console; and finally, the code is terminated satisfactorily.

```
        cout << "The number is less than 30!" << endl;
    }
    else
    {
        cout << "The number is greater than 30!" << endl;
    }
```

STEP 5

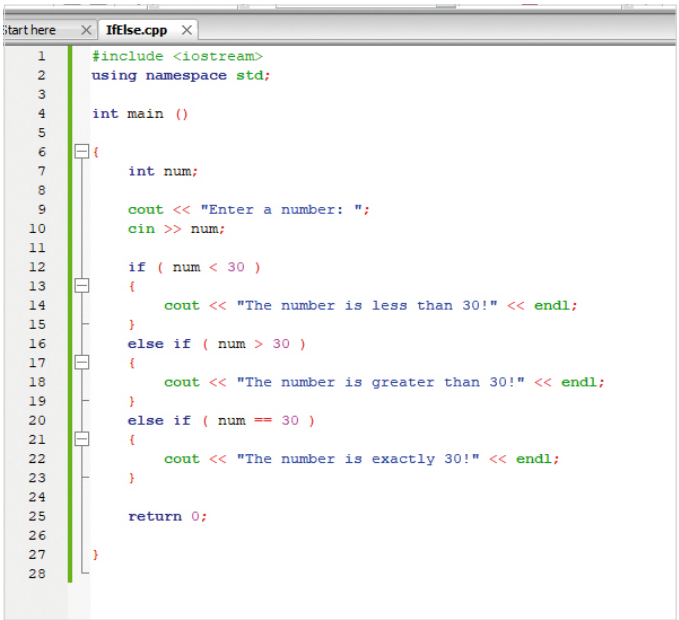
You can change the value of num in the code or you can improve the code by asking the user to enter a value:

```
{
    int num;

    cout << “Enter a number: “;
    cin >> num;

    if ( num < 30 )
    {
        cout << “The number is less than 30!” <<
endl;
    }
    else
    {
        cout << “The number is greater than 30!”
<< endl;
    }

    return 0;

}
```

STEP 6

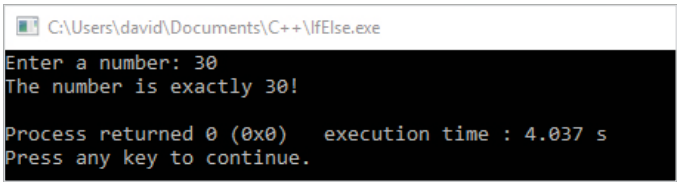
The code works the same way, as you would expect, but what if you wanted to display something if the user entered the number 30? Try this:

```
{
    int num;

    cout << “Enter a number: “;
    cin >> num;

    if ( num < 30 )
    {
        cout << “The number is less than 30!” <<
endl;
    }
    else if ( num > 30 )
    {
        cout << “The number is greater than 30!”
<< endl;
    }
    else if ( num == 30 )
    {
        cout << “The number is exactly 30!” <<
endl;
    }

    return 0;
}
```



STEP 7

The new addition to the code is what’s known as a nested if… else statement. This allows you to check for multiple conditions. In this case, if the user enters a number less than 30, greater than 30 or actually 30 itself, a different outcome is presented to them.



STEP 8

You can take this up a notch and create a two-player number guessing game. Begin by creating the variables:

```
int num, guess, tries = 0;

    cout << “***** Two-player number guessing game
****” << endl;
    cout << “\nPlayer One, enter a number for
Player Two to guess: “ << endl;
    cin >> num;
    cout << string(50, ‘\n’);
```

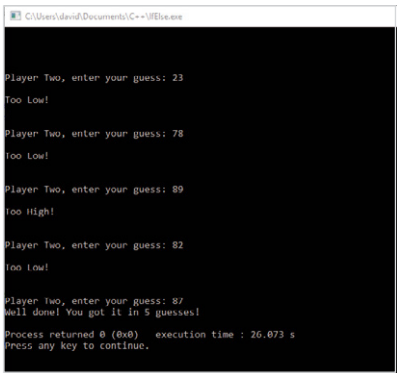
STEP 9

The cout << string(50, ‘\n’) line clears the screen so Player Two doesn’t see the entered number. Now you can create a do… while loop, together with if… else:

```
 do
    {
        cout << “\nPlayer Two, enter your guess: “;
        cin >> guess;
        tries++;
        if (guess > num)
        {
            cout << “\nToo High!\n” << endl;
        }
        else if (guess < num)
        {
            cout << “\nToo Low!\n” << endl;
        }
        else if (guess == num)
        {
            cout << “Well done! You got it in “ <<
tries << “ guesses!” << endl;
        }
    } while (guess != num);

    return 0;
```

STEP 10

Grab a second player, then build and run the code. Player One enters the number to be guessed, then Player Two can take as many guesses as they need to get the right number. Want to make it harder? Maybe use decimal numbers.



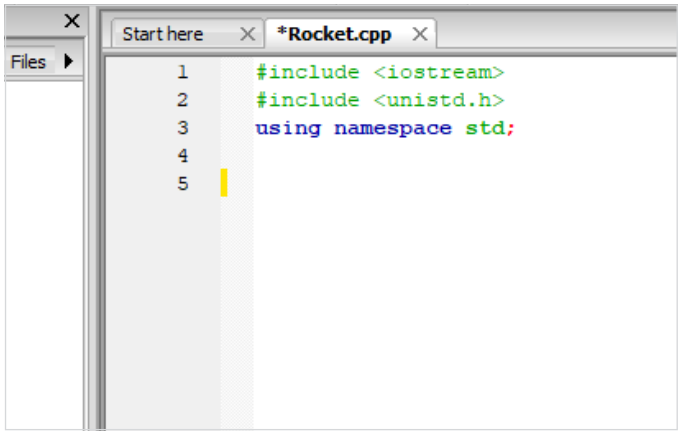
Combining All You Know

A good working knowledge of C++ can help engineer your future, regardless of whether your aim is to start a job as a developer, or simply to learn something new. The power of C++ is what makes it such a spectacular language to learn. Now, how about combining all you've learned so far.

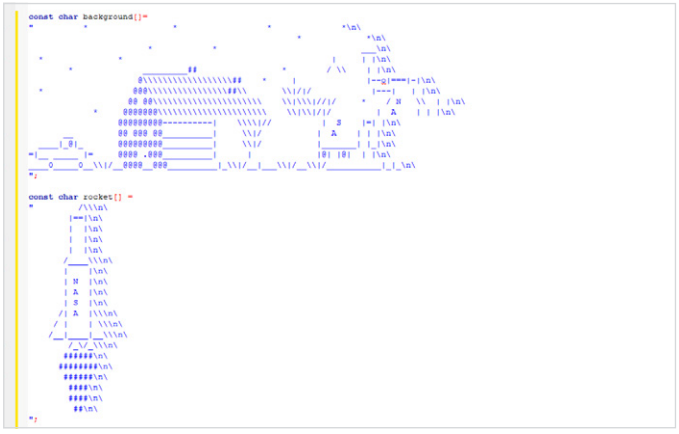
REACH FOR THE STARS

We're going to mix many elements of what we've looked at so far to create code that will animate a rocket blasting off into space. There's a lot than can be done with this code, so feel free to play with it.

STEP 1 Let's begin with a blank page. Enter the usual pre-processor, **#include <iostream>**, but also add a new one: **#include <unistd.h>** this is a pre-processor that provides access to the POSIX operating system API. In our case, we're going to use it to access a command called **Usleep**, as we'll see later.

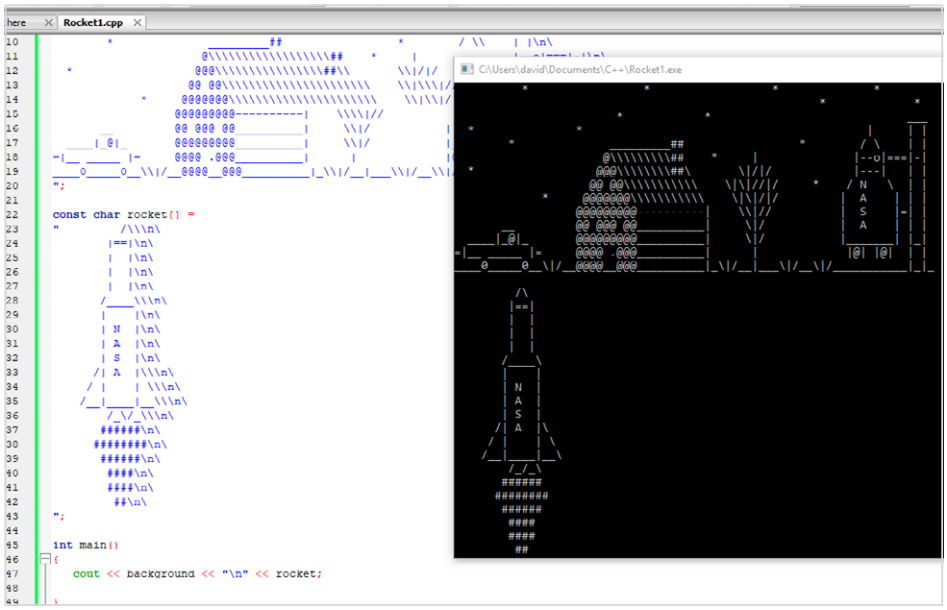


STEP 2 Next up, we're going to create a couple of data strings that will contain ASCII images that we can use in the code. You can of course create your own, if you're artistically inclined, however, if you're not, then take to Google and search for 'ASCII rocket art'. Define two strings, background and rocket.



STEP 3 The ASCII art looks messy to begin with. Remember though, you will need to enter a '\' after every '\' for it to be recognised and a '\n\' at the end of a line to signify a new line. It'll take some time to get right. You can test the results by entering:

```
Int main()
{
cout << background << "\n"
<< rocket;
{
```



STEP 4

When you've sorted the ASCII art out, remove the cout statements and enter the following:

```
string greet = “Are you ready to explore the stars?”;
    int x=0;

    cout << background;
    cout << “\n”;

    while (greet[x] !=’\0’)
    {
        cout << greet[x];
        usleep(95000);
        x++;
    }

    string name;
    cout << “\n” << “\n” << “What is your name,
       brave astronaut? “;
    cin >> name;
    cout << flush;
    cout << “Press Enter for take off, “ << name
       << “.\n”;
    cin.ignore();
    cin.get();
```

```
int main()
{
    string greet = "Are you ready to explore the stars?";
     int x=0;

     cout << background;
     cout << "\n";

     while (greet[x] !='\0')
     {
         cout << greet[x];
         usleep(95000);
         x++;
     }

     string name;
     cout << "\n" << "\n" << "What is your name, brave as
     cin >> name;
     cout << flush;
     cout << "Press Enter for take off, " << name << ".\n
     cin.ignore();
     cin.get();

}
```

STEP 5

What you've done here is create a simple greeting string, displayed the background ASCII art and asked the user for their name. Notice there's a While loop that will display the greet string, one letter at a time, until it reaches the end of the string. The **cout << flush** statement clears the screen and cin.get pauses, until the user hits Enter.

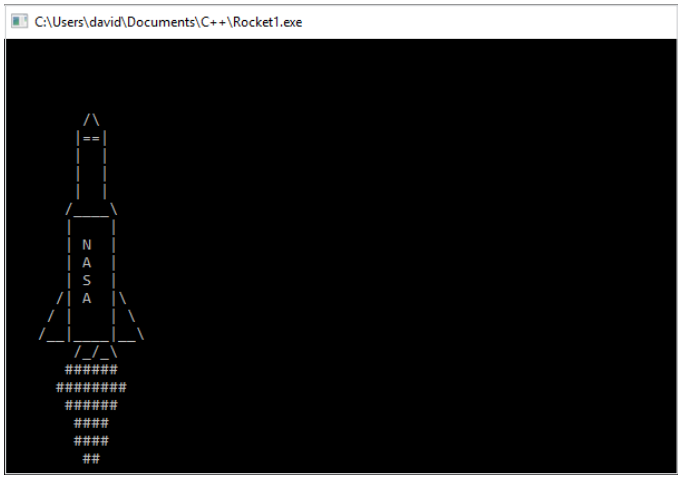
STEP 6

For the last section of the code, enter:

```
for (int i = 0; i < 50; i ++) cout << “\n”;
    cout << rocket;
    int j = 300000;
    for (int i = 0; i < 50; i ++)
    {
        usleep(j);
        j = (int)(j * 0.9);
        cout << “\n”;
    }

    cout << “Boldly go, “ << name << “!\n”;
    return 0;
{
```

```
for (int i = 0; i < 50; i ++) cout << "\n";
cout << rocket;
int j = 300000;
for (int i = 0; i < 50; i ++)
{
    usleep(j);
    j = (int)(j * 0.9);
    cout << "\n";
}
cout << "Boldly go, " << name << "!\n";
return 0;
```

STEP 7 The previous code displays and animates the rocket ASCII art. It uses a for loop to add an extra line under the rocket at increasing steps; using the Usleep function to lessen the amount of time between each new line, thus giving the appearance of the rocket speeding up into space. Finally, there's a message to the user at the end of the code.



STEP 8 Compile the code, ensuring that all brackets are closed off and there's a semicolon at the end of the appropriate lines. When you run through, you will be asked for your name, press Enter to clear the screen and watch the rocket animation (cheating animation, but animation all the same) take off, with a final 'Boldly go, NAME' message at the end.

IMPROVED GRAPHICS

Sadly, C++ in itself doesn't provide graphics. To get better graphics and animations, you will need to use some of the available libraries and full development engines. Much like Python, in some respects, you will need to apply the relevant extras (modules in the case of Python, libraries for C++) and learn how to code them into your own programs.

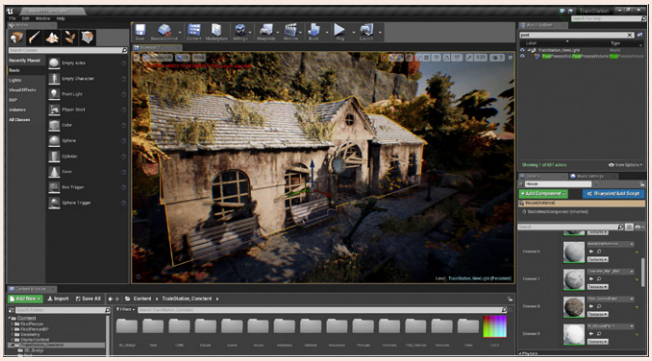
What you need to understand is that C++ development in the console, the command line if you will, is completely different to game development. Years ago, using Borland C++, you could easily utilise the Graphics.h library then display and animate EGA and VGA graphics.

If you're looking to stretch your C++ graphics and animations learning, then you're probably best learning the basics first, such as this section of the book, before moving into the more complex areas of C++. Then when you've got a grasp on the more advanced concepts of C++, the next step would be to get hold of a game engine, such as CryEngine, Unreal Engine, or AppGameKit. Each of these are used to create some of the most impressive games, with AppGameKit probably being the easiest of the bunch to begin with.

However to reiterate, as with most aspects of coding, building a good foundation first is essential before delving into the inner workings of a graphical or game engine.

With the correct tools, you can combine graphics and animations with C++ routines.

Game engines, together with C++ code, can be used to create anything from a top-selling game, to something personal.



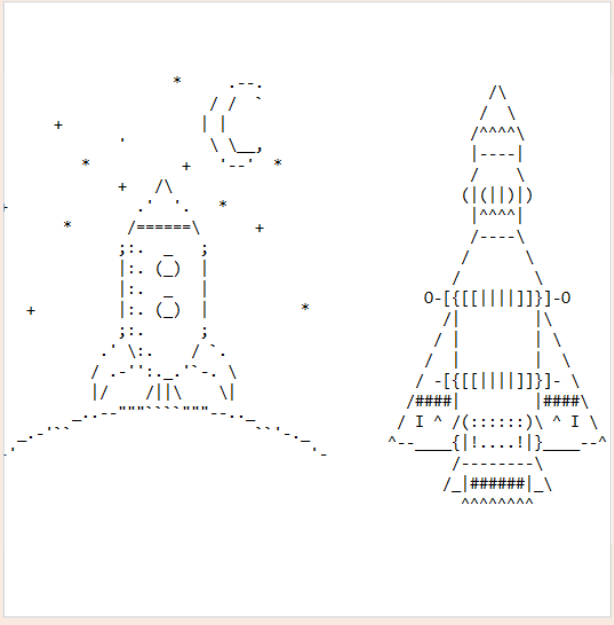
The Unreal Engine helps you develop rich environments and allows you to add C++ code behind the scenes.

STEP 9 Here's the code in its entirety:

```
#include <iostream>
#include <unistd.h>
using namespace std;

const char background[]=
“      *       *       *       *\n\
                                 *     *\n\
           *      *                    ___\n\
  *       *                    |      | |\n\
     *            _________##    *       / \\     | |\n\
                  @\\\\\\\\\\\\\\\\\\##     *      |
|--o|===|-|\n\
  *               @@@\\\\\\\\\\\\\\\\##\\        \\|/|/
|---|    | |\n\
                  @@ @@\\\\\\\\\\\\\\\\\\\\\\
\\|\\\\|//|/      *     / N    \\   | |\n\
        *      @@@@@@@\\\\\\\\\\\\\\\\\\\\\\\
\\|\\\|/|/           |  A     | | |\n\
                  @@@@@@@@@----------|     \\\\|//
|  S     |=| |\n\
         __           @@ @@@ @@__________|      \\|/
|  A     | | |\n\
  ____|_@|_         @@@@@@@@@__________|      \\|/
|_______| |_|\n\
=|__ _____ |=      @@@@ .@@@__________|       |
|@| |@|   | |\n\
____0_____0__\\|/__@@@@__@@@__________|_\\|/__|___
\\|/__\\|/___________|_|_\n\
“;

const char rocket[] =
“          /\\\n\
         |==|\n\
         |  |\n\
         |  |\n\
         |  |\n\
        /____\\\n\
        |    |\n\
        | N  |\n\
        | A  |\n\
        | S  |\n\
       /| A  |\\\n\
      / |    | \\\n\
     /__|____|__\\\n\
         /_\\/_\\\n\
        ######\n\
       ########\n\
        ######\n\
         ####\n\
         ####\n\
          ##\n\
“;

int main()
{
   string greet = “Are you ready to explore the
stars?”;
    int x=0;

    cout << background;
    cout << “\n”;
    while (greet[x] !=’\0’)
    {
        cout << greet[x];
        usleep(95000);
        x++;
    }

    string name;
    cout << “\n” << “\n” << “What is your name,
brave astronaut? “;
    cin >> name;
    cout << flush;
    cout << “Press Enter for take off, “ << name
<< “.\n”;
    cin.ignore();
    cin.get();

    for (int i = 0; i < 50; i ++) cout << “\n”;
    cout << rocket;
    int j = 300000;
    for (int i = 0; i < 50; i ++)
    {
        usleep(j);
        j = (int)(j * 0.9);
        cout << “\n”;
    }
    cout << “Boldly go, “ << name << “!\n”;

}
```

ASCII ART

The only major issue that can be time-consuming is getting the ASCII art to be displayed correctly. As you can see from the screenshots, it doesn't display correctly in the IDE, but when executed it displays perfectly fine in the console window. Take the time to play around with it, remember the slashes '\', and new lines, '\n\'. See what other art you can come up with.



"Controlling complexity is the essence of computer programming."

– Brian Kernighan (Co-developer of UNIX and author)

SHARE YOUR CODE!

The code listed within this section can be downloaded as Python files, so you don't have to type it out. Simply visit: www.bdmpublications.com/code-portal, sign up for access to the portal and the code is available as a compressed file for you to download and execute.

In addition, if you've written something amazing and want to show it off then why not send it in and we'll add it to the Code Portal as well as mention it via our social media accounts.

Tell us what the code does, how it works (don't forget to include comments in the code) and what platform to run it on.

Send it in to: enquiries:bdmpublications.com. We look forward to seeing what you've done.

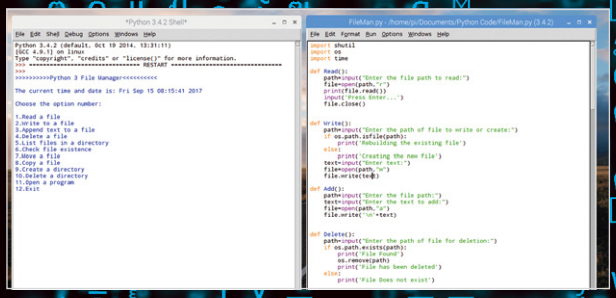
Code Repository

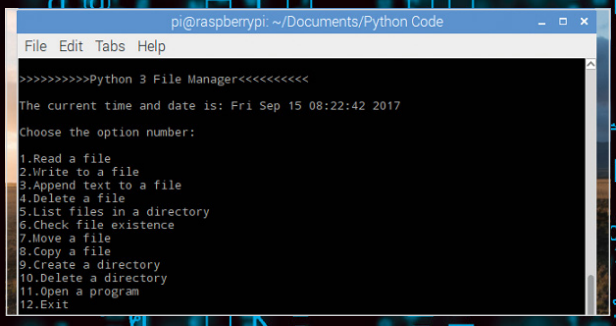
It's one thing to learn to code, but something else entirely to view a working program and be able to customise it for your own needs. With that in mind, we've got some great Python code examples available for you to try out and use.

There's code for a Python file manager, number guessing game, random password generator and even a game of Hangman. Use the code, tear it apart and improve it, it's free for you to play with.

Python File Manager

This file manager program displays a list of options that allow you to read a file, write to a file, append to a file, delete a file, list the contents of a directory and much more. It's remarkably easy to edit and insert into your own code, or add to.





1 This part of the code imports the necessary modules. The OS and Subprocess modules deal with the operating system elements of the program.

2 Each def XXX() functions store the code for each of the menu's options. Once the code within the function is complete, the code returns to the main menu for another option.

3 This is part of the code that checks to see what OS the user is running. In Windows the CLS command clears the screen, whereas in Linux and macOS, the Clear command wipes the screen. If the code tries to run CLS when being used in Linux or macOS, an error occurs, which then prompts it to run the Clear command instead.

4 These are the options, from 1 to 12. Each executes the appropriate function when the relevant number is entered.

FILEMAN.PY

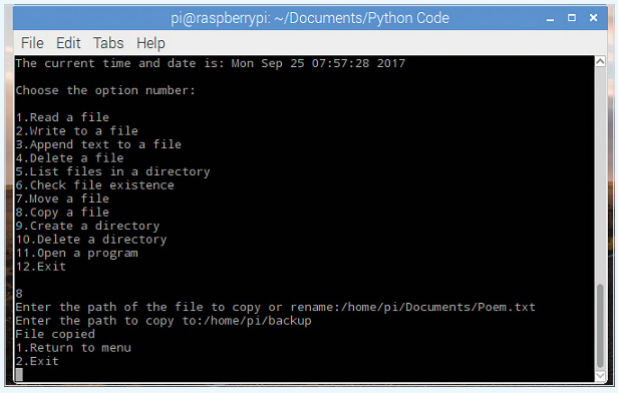
Copy the code below into a New > File and save it as FileMan.py. Once executed it will display the program title, along with the current time and date and the available options.

```
import shutil
import os
import time
import subprocess

def Read():
   path=input("Enter the file path to read:")
   file=open(path,"r")
   print(file.read())
   input('Press Enter...')
   file.close()

def Write():
   path=input("Enter the path of file to write or create:")
   if os.path.isfile(path):
      print('Rebuilding the existing file')
   else:
      print('Creating the new file')
   text=input("Enter text:")
   file=open(path,"w")
   file.write(text)

def Add():
   path=input("Enter the file path:")
   text=input("Enter the text to add:")
   file=open(path,"a")
   file.write('\n'+text)

def Delete():
   path=input("Enter the path of file for deletion:")
   if os.path.exists(path):
      print('File Found')
      os.remove(path)
      print('File has been deleted')
   else:
      print('File Does not exist')

def Dirlist():
   path=input("Enter the Directory path to display:")
   sortlist=sorted(os.listdir(path))
   i=0
   while(i<len(sortlist)):
      print(sortlist[i]+'\n')
      i+=1

def Check():
   fp=int(input('Check existence of \n1.File \n2.
   Directory\n'))
      if fp==1:
         path=input("Enter the file path:")
         os.path.isfile(path)
```

```
        if os.path.isfile(path)==True:
            print('File Found')
        else:
            print('File not found')
    if fp==2:
        path=input("Enter the directory path:")
        os.path.isdir(path)
        if os.path.isdir(path)==False:
            print('Directory Found')
        else:
            print('Directory Not Found')

def Move():
    path1=input('Enter the source path of file to move:')
    mr=int(input('1.Rename \n2.Move \n'))
    if mr==1:
        path2=input('Enter the destination path and file name:')
        shutil.move(path1,path2)
        print('File renamed')
    if mr==2:
        path2=input('Enter the path to move:')
        shutil.move(path1,path2)
        print('File moved')

def Copy():
    path1=input('Enter the path of the file to copy or rename:')
    path2=input('Enter the path to copy to:')
    shutil.copy(path1,path2)
    print('File copied')

def Makedir():
    path=input("Enter the directory name with path to make
    \neg. C:\\Hello\\Newdir \nWhere Newdir is new
    directory:")
    os.makedirs(path)
    print('Directory Created')

def Removedir():
    path=input('Enter the path of Directory:')
    treedir=int(input('1.Deleted Directory \n2.Delete
    Directory Tree \n3.Exit \n'))
    if treedir==1:
        os.rmdir(path)
    if treedir==2:
        shutil.rmtree(path)
        print('Directory Deleted')
    if treedir==3:
        exit()

def Openfile():
    path=input('Enter the path of program:')
    try:
        os.startfile(path)
    except:
        print('File not found')

run=1
while(run==1):
    try:
        os.system('clear')
    except OSError:
        os.system('cls')
    print('\n>>>>>>>>>>Python 3 File Manager<<<<<<<<<<\n')
    print('The current time and date is:',time.asctime())
    print('\nChoose the option number: \n')
    dec=int(input('''1.Read a file
2.Write to a file
3.Append text to a file
4.Delete a file
5.List files in a directory
6.Check file existence
7.Move a file
8.Copy a file
9.Create a directory
10.Delete a directory
11.Open a program
12.Exit

'''))
    if dec==1:
        Read()
    if dec==2:
        Write()
    if dec==3:
        Add()
    if dec==4:
        Delete()
    if dec==5:
        Dirlist()
    if dec==6:
        Check()
    if dec==7:
        Move()
    if dec==8:
        Copy()
    if dec==9:
        Makedir()
    if dec==10:
        Removedir()
    if dec==11:
        Openfile()
    if dec==12:
        exit()
    run=int(input("1.Return to menu\n2.Exit \n"))
    if run==2:
        exit()
```

3

4



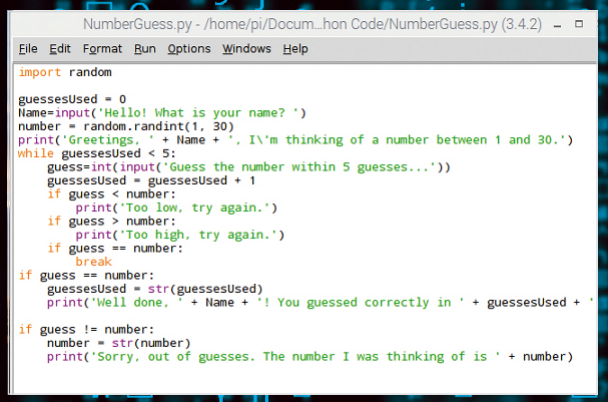
Imports

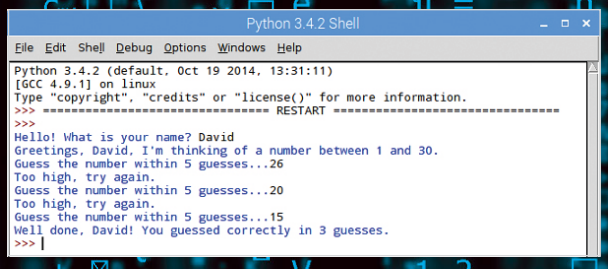
There are three modules to import here: Shutil, OS and Time. The first two deal with the operating system and file management and manipulation; and the Time module simply displays the current time and date.

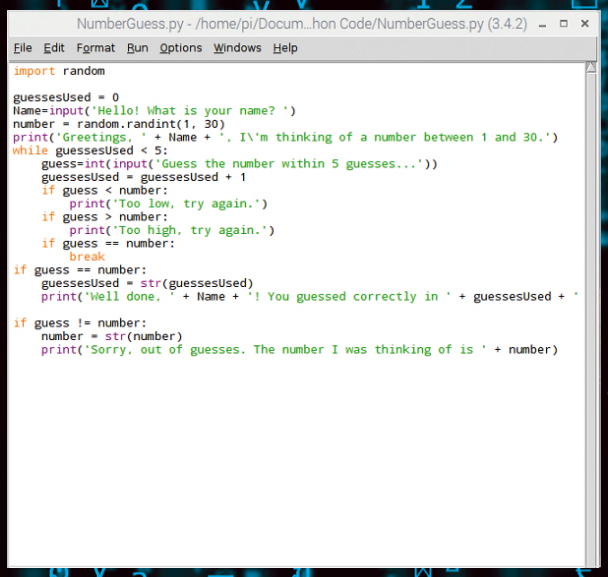
Note how we've included a try and except block to check if the user is running the code on a Linux system or Windows. Windows uses CLS to clear the screen, while Linux uses clear. The try block should work well enough but it's a point of possible improvement depending on your own system.

Number Guessing Game

This is a simple little piece of code but it makes good use of the Random module, print and input, and a while loop. The number of guesses can be increased from 5 and the random number range can easily be altered too.







NUMBERGUESS.PY

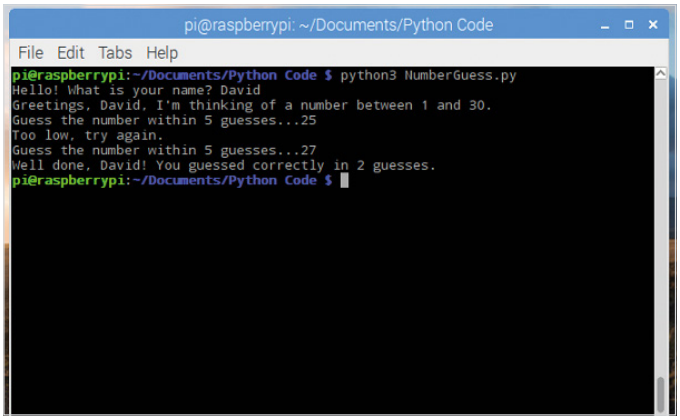
Copy the code and see if you can beat the computer within five guesses. It's an interesting bit of code that can be quite handy when your implementing a combination of the Random module alongside a while loop.

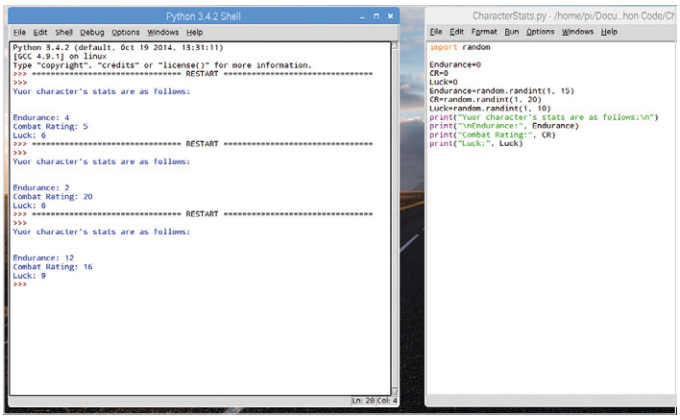
```
import random

guessesUsed = 0
Name=input('Hello! What is your name? ')
number = random.randint(1, 30)
print('Greetings, ' + Name + ', I\'m thinking of a
number between 1 and 30.')
while guessesUsed < 5:
   guess=int(input('Guess the number within 5 guesses...'))
   guessesUsed = guessesUsed + 1
   if guess < number:
      print('Too low, try again.')
   if guess > number:
      print('Too high, try again.')
   if guess == number:
      break
if guess == number:
   guessesUsed = str(guessesUsed)
   print('Well done, ' + Name + '! You guessed
   correctly in ' + guessesUsed + ' guesses.')

if guess != number:
   number = str(number)
   print('Sorry, out of guesses. The number I was
   thinking of is ' + number)
```

1 Although this is a reasonably easy to follow program, there are some elements to the code that are worth pointing out. To begin with, you need to import the Random module, as you're using random numbers within the code.

2 This section of the code creates the variables for the number of guesses used, along with the name of the player, and also sets up the random number between 1 and 30. If you want a wider range of random number selection, then increase the **number=random.randint(1, 30)** end value of 30; don't make it too high though or the player will never be able to guess it. If the player guesses too low or too high, they are given the appropriate output and asked to try again, while the number of guesses is less than five. You can also increase the number of guesses from 5 by altering the **while guessesUsed < 5:** value.

3 If the player guessed the correct number then they are given a 'well done' output, along with how many guesses they used up. If the player runs out of guesses, then the game over output is displayed instead, along with revealing the number the computer was thinking of. Remember, if you do alter the values of the random number chosen by the computer, or the number of guesses the player can take, then along with the variable values, you also need to amend the instructions given in the print statements at the start of the code.





Code Improvements

Since this is such as simple script to apply to a situation, there's plenty of room to mess around with it and make it more interesting. Perhaps you can include an option to take score, the best out of three rounds. Maybe an elaborate way to congratulate the player for getting a 'hole in one' correct guess on their first try.

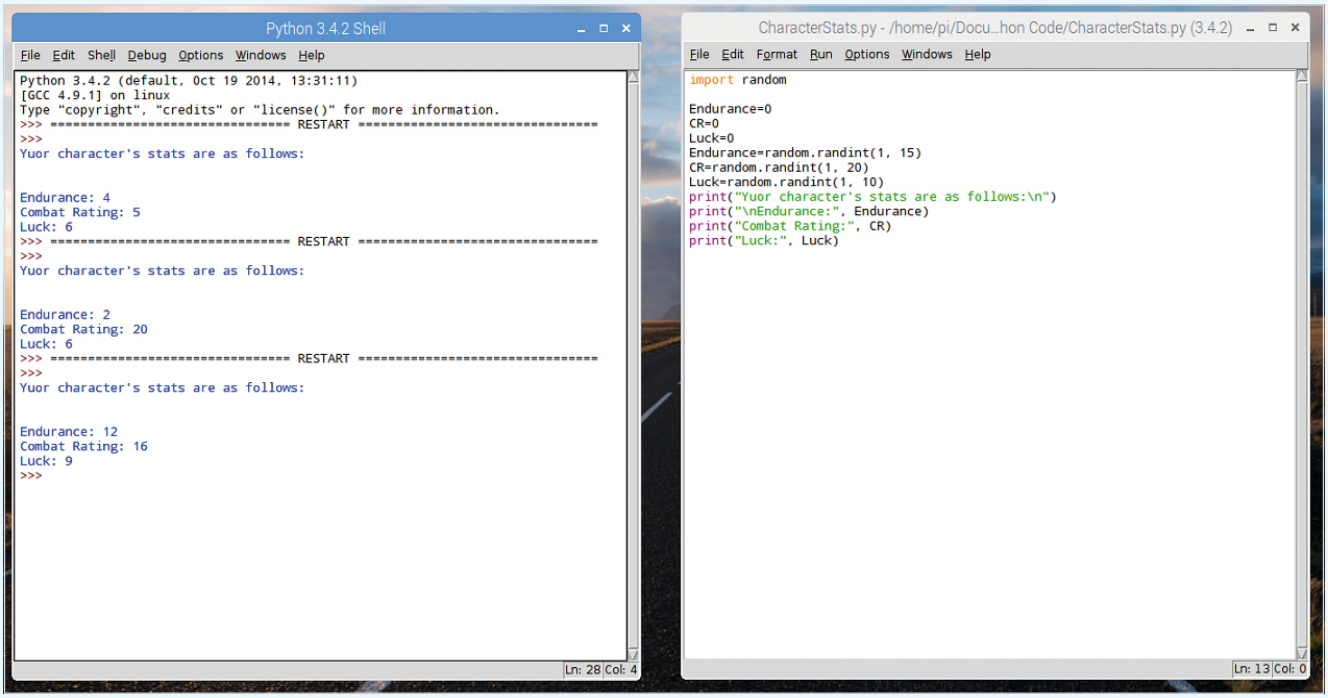
Moreover, the number guessing game code does offer some room for implementing into your code in a different manner. What we mean by this is, the code can be used to retrieve a random number between a range, which in turn can give you the start of a character creation defined function within an adventure game.

Imagine the start of a text adventure written in Python, where the player names their character. The next step is to roll the virtual random dice to decide what that character's combat rating, strength, endurance and luck values are. These can then be carried forward into the game under a set of variables that can be reduced or increased depending on the circumstances the player's character ends up in.

For example, as per the screenshot provided, you could use something along the lines of:

```
Endurance=0
CR=0
Luck=0
Endurance = random.randint(1, 15)
CR = random.randint(1, 20)
Luck = random.randint(1, 10)
Print("Your character's stats are as follows:\n")
Print("Endurance:", Endurance)
Print("Combat Rating:", CR)
Print("Luck:", Luck)
```

The player can then decide to either stick with their roll or try again for the hope of better values being picked. There's ample ways in which to implement this code into a basic adventure game.



Random Number Generator

User input and the ability to manipulate that input are important elements with any programming language. It's what separates a good program from a great program, one that allows the user to interact and see the results of that interaction.

RNDNUMGEN.PY

It might be simple but this little piece of code will ask the user for two sets of numbers, a start and a finish. The code will then pluck out a random number between the two sets and display it.

```
from random import *

print("\n>>>>>>>>>>Random Number Generator<<<<<<<<<<\n")
nmb1=int(input("Enter the start number: "))
nmb2=int(input("Enter the last number: "))

x = randint(nmb1, nmb2)
print("\nThe random number between",nmb1,"and",nmb2,"is:\n")
print(x)
```

More Input

While an easy code to follow, it could be more interesting if you prompt the user for more input. Perhaps you can provide them with addition, subtraction, multiplication elements with their numbers. If you're feeling clever, see if you can pass the code through some Tkinter GUI code, so it's presented in a window rather in the console.

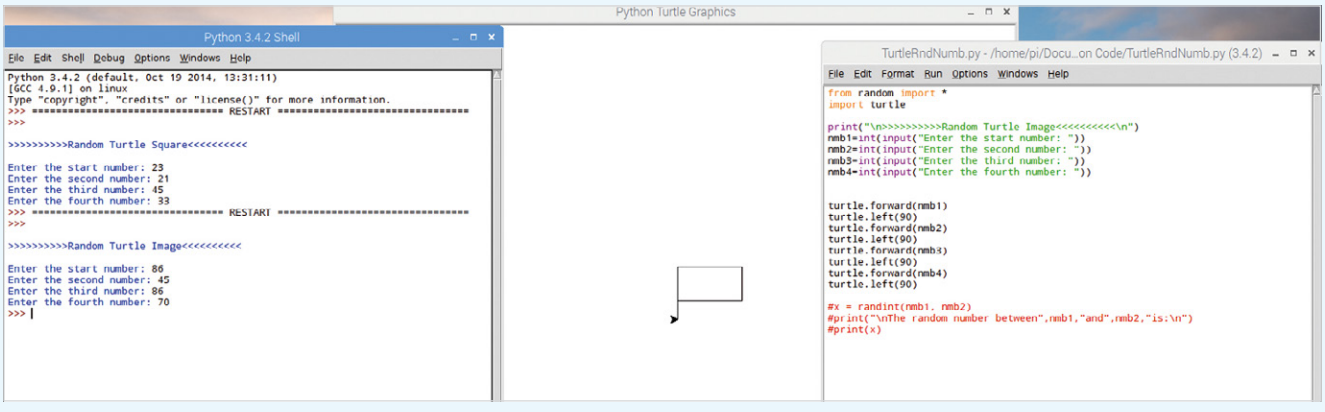
Furthermore, the core of the code can be used in a text adventure game, where the character fights something and their health, along with the enemy's, is reduced by a random number. This can be mixed with the previous code from Page 154's Number Guessing Game, where we defined the stats for the adventure game's character.

You can also introduce the Turtle module into the code and perhaps set some defined rules for drawing a shape, object or something based on a user inputted random value from a range of numbers. It takes a little working out but the effect is certainly really interesting.

For example, the code could be edited to this:

```
from random import *
import turtle

print("\n>>>>>>>>>>Random Turtle Image<<<<<<<<<<\n")
nmb1=int(input("Enter the start number: "))
nmb2=int(input("Enter the second number: "))
nmb3=int(input("Enter the third number: "))
nmb4=int(input("Enter the fourth number: "))


turtle.forward(nmb1)
turtle.left(90)
turtle.forward(nmb2)
turtle.left(90)
turtle.forward(nmb3)
turtle.left(90)
turtle.forward(nmb4)
turtle.left(90)
```

Whilst it's a little rough around the edges, you can easily make it more suitable.



Random Password Generator

We're always being told that our passwords aren't secure enough; well here's a solution for you to implement into your own future programs. The random password generator code below will create a 12-letter string of words (both cases) and numbers each time it's executed.

RNDPASSWORD.PY

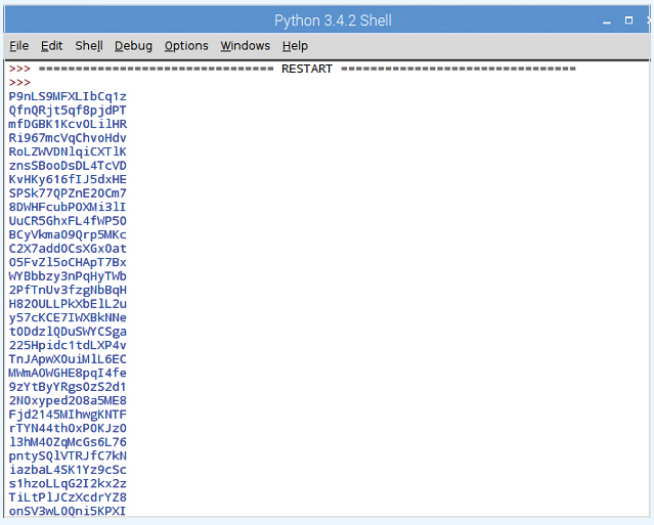
Copy the code and run it; each time you'll get a random string of characters that can easily be used as a secure password which will be incredibly difficult for a password cracker to hack.

```
import string
import random

def randompassword():
    chars=string.ascii_uppercase + string.ascii_
    lowercase + string.digits
    size= 8
    return ''.join(random.choice(chars) for x in
    range(size,20))

print(randompassword())
```

Secure Passwords

There's plenty you can do to modify this code and improve it further. For one, you can increase the number of characters the generated password displays and perhaps you can include special characters too, such as signs and symbols. Then, you can output the chosen password to a file, then securely compress it using the previous random number generator as a file password and send it to a user for their new password.

An interesting aspect to this code is the ability to introduce a loop and print any number of random passwords. Let's assume you have a list of 50 users for a company and you're in charge of generating a random password for them each month.



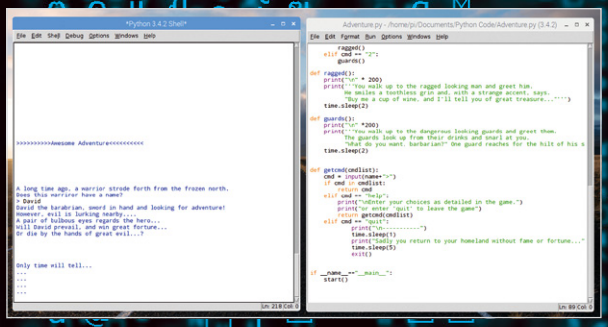
Adding a loop to print a password fifty times is extremely easy, for example:

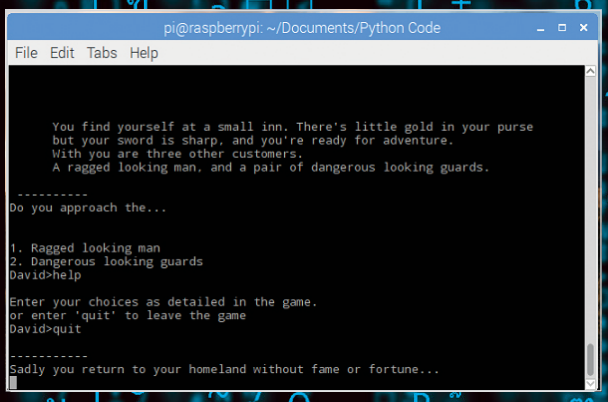
```
import string
import random


def randompassword():
    chars=string.ascii_uppercase + string.ascii_
    lowercase + string.digits
    size= 4
    return ''.join(random.choice(chars) for x in
    range(size,20))

n=0
while n<50:
    print(randompassword())
    n=n+1
```

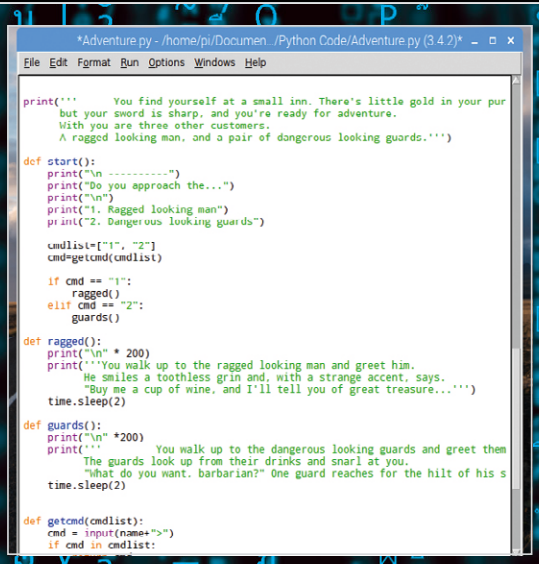
This will output fifty random passwords based on the previous random selection of characters.

Text Adventure Script

Text adventures are an excellent way to build your Python coding skills and have some fun at the same time. This example that we created will start you on the path to making a classic text adventure; where it will end is up to you.







ADVENTURE.PY

The Adventure game uses just the Time module to begin with, creating pauses between print functions. There's a help system in place to expand upon, as well as the story itself.

```
import time

print("\n" * 200)
print(">>>>>>>>>>Awesome Adventure<<<<<<<<<<\n")
print("\n" * 3)
time.sleep(3)
print("\nA long time ago, a warrior strode forth from
the frozen north.")
time.sleep(1)
print("Does this warrior have a name?")
name=input("> ")
print(name, "the barbarian, sword in hand and looking
for adventure!")
time.sleep(1)
print("However, evil is lurking nearby....")
time.sleep(1)
print("A pair of bulbous eyes regards the hero...")
time.sleep(1)
print("Will", name, "prevail, and win great fortune...")
time.sleep(1)
print("Or die by the hands of great evil...?")
time.sleep(1)
print("\n" *3)
print("Only time will tell...")
time.sleep(1)
print('...')
time.sleep(1)
print('...')
time.sleep(1)
print('...')
time.sleep(1)
print('...')
time.sleep(5)
print("\n" *200)

print('''      You find yourself at a small inn. There's
   little gold in your purse but your sword is sharp,
   and you're ready for adventure.
   With you are three other customers.
   A ragged looking man, and a pair of dangerous
   looking guards.''')

def start():
   print("\n ----------")
   print("Do you approach the...")
   print("\n")
   print("1. Ragged looking man")
   print("2. Dangerous looking guards")

   cmdlist=["1", "2"]
   cmd=getcmd(cmdlist)
```

```
    if cmd == "1":
        ragged()
    elif cmd == "2":
        guards()

def ragged():
    print("\n" * 200)
    print('''You walk up to the ragged looking man and
    greet him.
        He smiles a toothless grin and, with a strange
        accent, says.
        "Buy me a cup of wine, and I'll tell you of
        great treasure...'''')
    time.sleep(2)

def guards():
    print("\n" *200)
    print('''You walk up to the dangerous looking guards
    and greet them.
        The guards look up from their drinks and
        snarl at you.
        "What do you want, barbarian?" One guard reaches
        for the hilt of his sword...'''')
    time.sleep(2)
```

```
def getcmd(cmdlist):
    cmd = input(name+">")
    if cmd in cmdlist:
        return cmd
    elif cmd == "help":
        print("\nEnter your choices as detailed in
        the game.")
        print("or enter 'quit' to leave the game")
        return getcmd(cmdlist)
    elif cmd == "quit":
        print("\n----------")
        time.sleep(1)
        print("Sadly you return to your homeland without
        fame or fortune...")
        time.sleep(5)
        exit()


if __name__=="__main__":
    start()
```

Adventure Time

This, as you can see, is just the beginning of the adventure and takes up a fair few lines of code. When you expand it, and weave the story along, you'll find that you can repeat certain instances such as a chance meeting with an enemy or the like.

We've created each of the two encounters as a defined set of functions, along with a list of possible choices under the cmdlist list, and cmd variable, of which is also a defined function. Expanding on this is quite easy, just map out each encounter and choice and create a defined function around it. Providing the user doesn't enter quit into the adventure, they can keep playing.

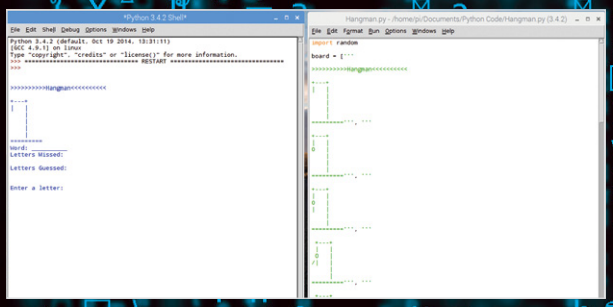
There's also room in the adventure for a set of variables designed for combat, luck, health, endurance and even an inventory or amount of gold earned. Each successful combat situation can reduce the main character's health but increase their combat skills or endurance. Plus, they could loot the body and gain gold, or earn gold through quests.

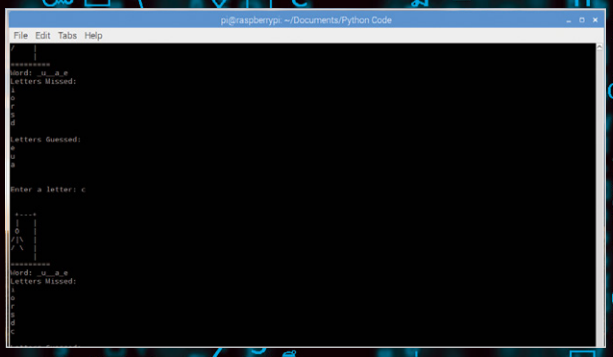
Finally, how about introducing the Random module. This will enable you to include an element of chance in the game. For example, in combat, when you strike an enemy you will do a random amount of damage as will they. You could even work out the maths behind improving the chance of a better hit based on your or your opponent's combat skills, current health, strength and endurance. You could create a game of dice in the inn, to see if you win or lose gold (again, improve the chances of winning by working out your luck factor into the equation).

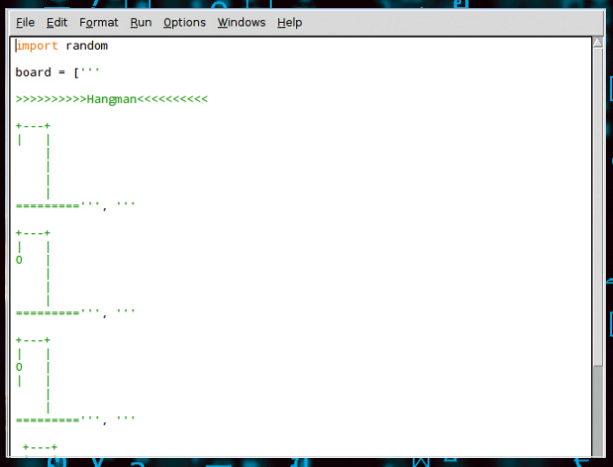
Needless to say, your text adventure can grow exponentially and prove to be a work of wonder. Good luck, and have fun with your adventure.

Hangman Game Script

Hangman is a great game to program into Python. It can be extremely complex, displaying graphics, the number of guesses left in the secret word, a huge bank of available words picked at random and countless other elements. It can also be quite simple. Here we have a mix between the two.







HANGMAN.PY

We've made a Hangman game board (the gallows) out of characters that can be displayed in the IDLE Shell, along with a huge bank of words to randomly choose from.

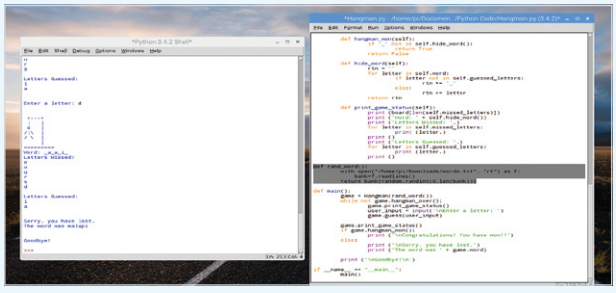
```
import random

board = ['''

>>>>>>>>>>Hangman<<<<<<<<<<

+---+
|   |
    |
    |
    |
    |
=========''', '''

+---+
|   |
O   |
    |
    |
    |
=========''', '''

+---+
|   |
O   |
|   |
    |
    |
=========''', '''

 +---+
 |   |
 O   |
/|   |
     |
     |
=========''', '''

 +---+
 |   |
 O   |
/|\  |
     |
     |
=========''', '''

 +---+
 |   |
 O   |
/|\  |
/    |
```

```
        |
=========''', '''

 +---+
 |     |
 O     |
/|\    |
/ \    |
        |
=========''']


class Hangman:
   def __init__(self,word):
      self.word = word
      self.missed_letters = []
      self.guessed_letters = []

   def guess(self,letter):
      if letter in self.word and letter not in self.
      guessed_letters:
         self.guessed_letters.append(letter)
      elif letter not in self.word and letter not in
      self.missed_letters:
         self.missed_letters.append(letter)
      else:
         return False
      return True

   def hangman_over(self):
      return self.hangman_won() or (len(self.missed_
      letters) == 6)

   def hangman_won(self):
      if '_' not in self.hide_word():
         return True
      return False

   def hide_word(self):
      rtn = ''
      for letter in self.word:
         if letter not in self.guessed_letters:
            rtn += '_'
         else:
            rtn += letter
      return rtn

   def print_game_status(self):
      print (board[len(self.missed_letters)])
      print ('Word: ' + self.hide_word())
      print ('Letters Missed: ',)
      for letter in self.missed_letters:
         print (letter,)
      print ()
      print ('Letters Guessed: ',)
      for letter in self.guessed_letters:
         print (letter,)
      print ()

def rand_word():
   bank = 'ability about above absolute accessible
   accommodation accounting beautiful bookstore
   calculator clever engaged engineer enough
   handsome refrigerator opposite socks interested
   strawberry backgammon anniversary confused
   dangerous entertainment exhausted impossible
   overweight temperature vacation scissors
   accommodation appointment decrease development
   earthquake environment brand environment necessary
   luggage responsible ambassador circumstance
   congratulate frequent'.split()
   return bank[random.randint(0,len(bank))]

def main():
   game = Hangman(rand_word())
   while not game.hangman_over():
      game.print_game_status()
      user_input = input('\nEnter a letter: ')
      game.guess(user_input)

   game.print_game_status()
   if game.hangman_won():
      print ('\nCongratulations! You have won!!')
   else:
      print ('\nSorry, you have lost.')
      print ('The word was ' + game.word)

   print ('\nGoodbye!\n')

if __name__ == "__main__":
   main()
```

QUIT()

Since this is the last example in our Python code repository, we thought we'd go out with a bang and feature the hangman gallows being drawn with each incorrect guess of the word. Don't worry if it looks misaligned in the text here, this is merely due to the differences between using the Python IDLE editor and pasting the code into a word processor (which formats things differently).

There's plenty you can do to improve, enhance and expand on what we've presented here. You can include a routine that returns an error if the user enters a number or character. You can include extra points for someone who guesses the entire word in one go rather than one letter at a time and you could perhaps add Chopin's Funeral March should you lose the game; or something celebratory if you win.



Consider replacing the bank of words too. They're found under the bank list, and could easily be swapped out for something more difficult. If you download **www.github.com/dwyl/english-words** you can find a text document with over 466,000 words. Perhaps you could swap the words in the bank to instead read the contents of the text file:

```
def rand_word():
   with open("/home/pi/Downloads/words.txt", "rt") as f:
      bank=f.readlines()
   return bank[random.randint(0,len(bank))]
```

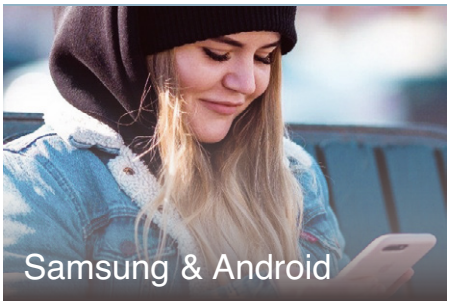
Black Dog Media

Master Your Tech — From Beginner to Expert

To continue learning more about your tech visit us at:

www.bdmpublications.com

FREE Tech Guides



EXCLUSIVE Offers on our Tech Guidebooks

- Print & digital editions
- Featuring the very latest updates
- Step-by-step tutorials and guides
- Created by BDM experts

Check out our latest titles today!

bdmpublications.com/ultimate-photoshop

Buy our Photoshop guides and download tutorial images for free!

Simply sign-up and get creative.

PLUS

Special Deals and Bonus Content

Sign up to our monthly newsletter and get the latest updates, offers and news from BDM. We are here to help you Master Your Tech!

The Ultimate Jargon Buster

Avoid tech confusion, either when reading this book or when talking to friends, with this glossary of technology terms and phrases. We have looked across the tech boarders to bring to you the definitive jargon buster, but it should help you to understand the common terms people use when talking about their devices and their software.

0-9

3G
The third generation of mobile data networking used by both the iPhone and iPad. This connection is slower than Wi-Fi, but is more readily available and is used to transfer data from your device when you are on the go. It uses the mobile phone network.

4G
The fourth generation of mobile data networking.

5G
The fifth generation of mobile data networking offers increased speed when transferring data on the go but it is still in its early stages of adoption by mobile phone networks.

A...

Accessibility
A series of tools and features designed to make an Apple device such as the Mac and mobile devices easier to use by those with disabilities such as vision or hearing impairments. You can find the Mac's Accessibility features and customise them in System Preferences.

ADB
Android Debug Bridge. Part of the Android Software Development Kit, used to send commands from a computer to an attached phone.

Adobe Bridge
Bridge is a browser application produced by Adobe Systems as part of the Creative Suite and is usually installed alongside Photoshop. Its main function is as the file management hub of the Creative Suite. It can be used to open, manage, rate and rename files as well as edit their metadata.

Adobe RGB
A device independent colour space developed by Adobe. It provides a relatively large range of colours, i.e. grey-balanced and perceptually uniform. It is widely used for image editing.

ADSL
Asymmetric Digital Subscriber Line. It's a means of connecting to the Internet through your telephone line. Sometimes just called 'DSL'

Airplane Mode
All airlines warn you to turn off mobile electronic devices when on board an aircraft, so this iPad setting turns off all incoming and outgoing signals to your device, including data, Bluetooth and Wi-Fi.

AirPlay
A protocol for streaming sounds and video from an Apple device to a set of compatible speakers or a device such as an Apple TV. It's wireless and easy to use.

AMOLED
Active Matrix Organic Light Emitting Diode. A bright and colourful display technology popular on smartphones (although it has now been superseded by Super AMOLED and qHD.)

Android
The name of the operating on your smartphone (we are assuming you own an Android phone if you are reading this magazine). There have so far been eleven versions/updates released.

Android Market
The previous name for the Google Play Store. The place to go to find apps, books and movies to install on your phone.

Anti-Aliasing Filter
This is an optical filter, also known as low-pass filter, which is placed on the camera sensor to create a slight blur or softening that helps counteract aliasing or Moiré interference.

Apk (.apk)
The file extension of Android applications.

Apps (Applications)
The programs, such as Angry Birds, Facebook or Soundhound, that you install and run on you Android phone.

App Store
The App Store is where you can download free and paid programs to your device using your Apple ID. You can access it through the application found on your home screen.

App Inventor
A web-based system that lets anyone develop apps for Android. Originally created and run by Google, but now run as an open-source project.

Apple ID
This is the email address and password that you have registered with Apple. It will be required to access most online applications on your iPad, including iTunes, App Store and Books.

Apple Menu
The menu that's opened by clicking on the Apple icon in the left of the menu bar, when using a Mac or MacBook computer. It gives access to system functions such as Preferences, App Store, Force Quit and more.

Archos
A manufacturer of Android tablets.

ASUS
A well-known manufacturer of Android smartphones and tablets.

AT&T
One of the "Big Four" of American carriers

B...

Bit
A contraction of binary digit, the smallest unit of information storage or digital information that can take on one of two values, 0 and 1.

Bit Depth
Defines how many bits of colour data are used to describe each pixel or channel. For example, 2 bits per pixel only allows for black or white. 8 bits provides 256 colours. When referring to an 8-bit colour image, 256 is multiplied by the three primary channels (red, green and blue) to create what is commonly called 24-bit colour, with a possible 16,777,266 colours.

Black Point
In image editing, the black point is a tonal adjustment that sets the point at which the deepest shadow detail in the histogram is clipped to black.

Bloatware
The name given to unwanted applications preloaded onto your phone. Bloatware cannot usually be removed by the end user unless they decide to root their handset.

BlueTooth
Short range file data system built into almost every Android smartphone ever made. Can be used to send files and connect speakers or headphones wirelessly to your phone.

Books
Apple's eBook reader, available from the App Store. It handles the standard electronic publishing formats protected by FairPlay DRM and PDF. It was introduced in 2010 along with the iPad.

Bootloader
A normally hidden mode in Android that helps with flashing ROMs when rooting an Android phone.

Broadband
Wide bandwidth data transmission, that is, fast Internet as opposed to the older, dial-up services.

Browser
An app used to access websites found on the worldwide web. The iPad and iPhone come with Apple's Safari browser preinstalled but others are available in the App Store. Android devices use the Chrome browser

BSI
Backside Illumination. Sometimes used to improve smartphone camera performance.

C...

Calendar
This is one of several preloaded apps found on most devices. Use it to keep track of events, invitations and reminders on your phone and tablet.

Camera Raw
Proprietary raw file formats designed to hold image data and metadata generated by digital cameras. These formats are non-standard and undocumented, although they are usually based on the TIFF/EP file format standard.

Carrier
Another name for a mobile network provider (Vodafone, AT&A, Sprint, etc.)

Casting
The process of converting one data-type into another. For example, sometimes a number may stored as text but need to be converted in to an integer.

CCD (Charged Coupled Device)
A type of image sensor found in digital cameras and scanners. It is a light-sensitive chip that converts light into an electrical charge that is then processed by an analogue to digital converter. CCD differs from the other common sensor type (CMOS) in the way that it processes the electrical charges captured by sensor elements.

CDMA
One of the two main cell phone communication standards. Not often used in phones outside of the U.S.

Chromatic Aberration
Known also as colour fringing, chromatic aberration is caused when a camera lens does not focus the different wavelengths of light onto the exact same focal plane. The effect is visible as a thin coloured halo around objects in the scene, often the border between dark and light objects.

Class
A class provides a means of bundling data and functionality together. They are used to encapsulate variables and functions into a single entity.

Clipping
The loss or either highlight or shadow details when tone information is forced to pure white or black. For example, over-exposure can produce clipping by forcing highlights that should contain detail to register as pure white. Clipping can also be caused either intentionally as a creative effect or unintentionally because of excessive corrections. Saturation clipping can occur when colours are pushed beyond the range of a colour space.

Comments
A comment is a section of real world wording inserted by the programmer to help document what's going on in the code. They can be single line or multi-line and are defined by a # or '".

Constant
A number that does not change. It is good practice to name constants in capitals e.g. SPEED_OF_LIGHT

CMOS (Complementary Metal Oxide Semiconductor)
A type of image sensor found in digital cameras and scanners. It is a light-sensitive chip that converts light into an electrical charge, which is then processed by an analogue to digital converter. CMOS differs from the other common sensor type (CCD) in the way that it processes the electrical charges captured by sensor elements.

CMYK
Also commonly referred to as process colour, CMYK is a subtractive colour model using cyan, magenta, yellow and black inks in colour printing.

Colour Profile
Also known as an ICC profile, the Colour Profile defines the information required to by a colour management system (CMS), to make the colour transformations between colour spaces. They can be device specific such as monitors, scanners or printers or abstract editing spaces.

Compression
The process of re-encoding digital information using fewer bits than the original file or source. This reduces transmission time and storage requirements. There are a number of different algorithms that provide either "lossy" or lossless compression. JEPG is a common file format that employs lossy compression to achieve smaller file sizes at the expense of image quality.

Cupcake
The nickname for Android version 1.5.

CyanogenMod
One of the best known and most often used series of custom ROMs.

D...

DECT
Digital Enhanced Cordless Telecommunications. It's a wireless standard used mostly for cable-free telephone systems.

DLNA
Dynamic Living Network Alliance. A technology found on some high-end Android phones that lets users stream photos and videos from their phone to a compatible TV.

DNG (Digital Negative)
An open standard file format developed by Adobe Systems that provides an alternative to proprietary camera raw files. The DNG specification incorporates rich metadata along with embedded previews, camera profiles and editable notes. DNG uses lossless compression that can result in a significant file size reduction over the original proprietary raw.

Download
The term used when taking a file from the Internet or from a connected device such as a computer, to your phone or tablet.

Dock
The opaque strip at the bottom of the home screen. Apps in the dock remain in a special row of icons (or Folders post iOS 4) along the bottom of iPhone, iPod touch and iPad screens and do not change when you swipe between home screens.

DPI (Dots Per Inch)
The measurement of print resolution expressed in how many dots of ink are laid down either horizontally or vertically per inch. A higher number indicates a greater amount of output resolution. Not to be confused with pixel per inch (PPI). There is not necessarily a direct correlation between DPI and PPI.

Dream (HTC Dream or G1)
The very first phone to use the Android operating system.

Dynamic Range
In the context of photography, dynamic range describes the difference between the brightest and darkest light intensities of a scene. From capture to output, there can be a large difference in the size of the dynamic range that each device is capable capturing or reproducing. Dynamic range is commonly expressed in the number of f-stops that can be captured or the contrast ratio of the scene or device.

E...

Eclair
The nickname for Android version 2.0/2.1.

Emoticon
A small drawing used to augment a message or text. Typically these are yellow faces showing a variety of expressions.

Escape Sequence
When characters that have certain meanings in the Python coding language are required in strings they have to be "escaped" so that the computer knows they do not have their usual meaning. This is done by putting a slash in front of them e.g. \"

Ethernet
The format used for local cabled networks (LAN). Your router comes supplied with Ethernet cables and has ports for plugging them in.

Exposure
The total amount of light that strikes the sensor or film during an image capture. An optimal exposure takes full advantage of the dynamic range of the sensor without under-exposing the shadows or over-exposing the highlights.

Extender
A device that extends the range of a wireless network by creating a second entry point, which may, or may not, merge with the main one.

F...

Facebook
Currently the most popular social networking site on the Internet; there are currently over 835 million registered users.

FaceTime
Apple's video calling service. Requires a Wi-Fi connection and is currently only supported via a phone number on iPhone and Apple ID email address on iPod touch 4 and Mac.

Factory Reset
An option on your Android phone that allows you to return it to the state it was when it left the factory.

File Format
The structure of how information is encoded in a computer file. File formats are designed to store specific types of information, such as JPEG and TIFF for image or raster data, AI for vector data or PDF for document exchange.

Folder
An icon representing a container for a group of apps, files or icons.

Force Quit
In the Fast App Switcher, tapping and holding an app will put it in 'jiggly mode' and tapping the x badge will force it to quit. Built-in apps like Mail and Messages will automatically restart while third-party apps will restart the next time you launch them.

Froyo
The nickname given to Android version 2.2.

G...

G1
The very first phone to run the Android operating system. Also known and the HTC Dream.

Game Center
Apple's gaming service, where you can discover new games and share your game experiences with friends from around the world.

Gamut
The range of colours and tonal values that can be produced by a capture or output device or represented by a colour space.

Galaxy
A range of hugely popular handsets from Samsung, the biggest smartphone manufacturer in the world.

Geotagging
The act of digitally attaching your location to photos taken on your phone.

Gingerbread
The nickname given to Android version 2.3.

Gmail
Google's web-based email software. Comes pre-installed on every Android smartphone.

Google
Owner (although not the original creator) of Android. Also own a fairly well known search engine...

Google Now
An enhanced Google search app which bases the information displayed on current location. Currently only found in Jelly Bean.

Google Play
Previously known as Android Market, this is where you go to download Android compatible apps, books, music and movies.

Gorilla Glass
Increasingly popular scratch-resistant glass used for smartphone displays.

GPS
Global Positioning System. A system that uses satellites to pinpoint your current location.

Grayscale
A monochromatic digital image file with pixel values that use shades of grey to represent tonal information. The term is often used to describe digital black and white photographs.

GSM
One of the two main cell phone communication standards. Used in most countries outside of the U.S.

H...

Hacking
Most often means rooting when talking about Android.

Hard Reset
Also called Factory Reset. Returns the phone to its post-factory state.

HDR (High Dynamic Range)
A process that combines multiple exposure variations of an image to achieve a dynamic range exceeding that of a single exposure. Algorithms are used to blend the exposures into a high-bit file format that can then be converted to either 8 or 16 bit for printing or web presentation.

Histogram
A graphical representation of the tone and colour distribution in a digital image. This is typically based on a particular colour or working space by plotting the number of pixels for each tone or colour value. It can be used to interpret photographic exposure and reveal shadow or highlight clipping.

Home Button
The physical hardware button on the front of early models of the iPhone, iPod touch, iPad and many Android devices, located just below the screen. It's used to wake the device, return to the Home Screen and several other functions.

Home Screen
The front end of your smartphone or tablet. The screen you see, containing app icons, widgets, etc., when you first unlock the device.

Honeycomb
The nickname given to Android version 3.0. The only version designed specifically for tablets, but now superseded by ICS.

HTC
A large Taiwanese smartphone manufacturer.

HTTP
Hypertext Transfer Protocol, the protocol used by the World Wide Web (Internet) that defines how messages are sent, received and read by browsers and other connected software layers.

HTTPS
Hypertext Transfer Protocol Secure, an encrypted and far more secure version of HTTP.

I...

Ice Cream Sandwich
The nickname given to Android version 4.0/4.1. The majority of new Android tablets now use this.

IMEI
International Mobile Equipment Identity. This is a unique identification number assigned to every phone.

Intel
Well known PC processor manufacturer. Has now started producing smartphone processors.

Internet
A global system of interconnected computers and networks which use the Internet Protocol Suite (TCP/IP) to link online devices.

Indentation
The coding language Python uses indentation to delimit blocks of code. The indents are four spaces apart, and are often created automatically after a colon is used in the code

iOS
Apple mobile operating system and the software that powers the iPhone, iPod touch, iPad and Apple TV.

IPS
In Plane Switching is a type of display used on some phones that increases the viewing angle of the screen.

I-Tunes
Mac and Windows music playing software, also used to activate and sync iPhone, iPod touch and iPad. It is also used to purchase and manage music, movies, TV shows, apps, books and other media.

ISO (International Organisation for Standardisation)
In photography, ISO refers to the standard for measurement of the sensitivity of film or digital sensors to light.

J...

Jelly Bean
The nickname given to Android 4.2, the latest version of the operating system.

JIT
The Just In Time compiler was introduced in Android 2.2. It helps to speed up apps on Android.

JPEG, JPG (Joint Photographic Experts Group)
A standard created by the Joint Photographic Experts Group for the compression of photographic images and the accompanying file format. It employs lossy compression that can reduce file size but at the expense of image quality and detail.

K...

Kernel
The basic Linux building block of Android.

Keyboard
Tablets and smartphones can feature either a physical or software keyboard.

Keyword
An element of metadata that is used to make a file more easily discoverable to searches. Keywords can be individual words or short phrases and can have a hierarchical structure.

L...

Landscape Mode
This describes a phone or tablet when you hold it horizontally; this is when it's wider than it is tall and the Home button is on the right or left of the screen.

Launcher
This is the part of the Android user interface on home screens that lets you launch apps and make phone calls.

LAN
Local Area Network. Devices that are connected to your router using Ethernet cables, are part of the LAN (see also WLAN).

LG
A large Korean electronics and smartphone manufacturer.

Linux
An open-source operating system that is used as the basis of Android.

Live Wallpapers
Animated wallpapers introduced in Android 2.1.

Loop
A piece of code that repeats itself until a certain condition is met. Loops can encase the entire code or just sections of it.

LTE
Long-Term Evolution. A name sometimes given to 4G data networks.

Luminance
The intensity of light as emitted or reflected by an object or surface. This is usually expressed in candelas per square meter (cd/m2). It is a measurement of the brightness of an object or light source.

M...

Magic
HTC phone also known as the MyTouch 3G. The first phone to use an Android operating system.

Mail
Built-in Apple app for handling POP3, IMAP, MobileMe and Exchange/ActiveSync email accounts.

Messages
One of Apple's built-in iPhone apps that handles SMS text messages and MMS multimedia messages. SMS messages are also more generally called "messages" on most devices.

MMS: (MultimediaMmessages)
MMS supports images, videos, sound, contact cards and location data. Sent and received via the Messages app.

Megapixel
A term used to describe digital camera resolution, 1 megapixel equals one million pixels or sensor elements. To calculate the megapixel value for a camera, multiply the horizontal by the vertical pixel counts of the recorded image.

Mesh
A means of combining two wireless access points into one, so they use the same settings and appear as a single network to devices that join it.

Metadata
Embedded or associated information describing a file's contents, used in digital photography to hold exposure information, GPS location data, copyright information and more. There are several metadata formats such as EXIF, IIM, IPTC Core, Dublin Core, DICOM and XMP.

Modem
Short for modulate-demodulate, a modem converts data into a signal that can be transferred over a phone line, and does so in reverse for incoming data.

Motorola
A large manufacturer or electronics and smartphones.

N...

News
Is an app in iOS that collected together magazine and newspaper apps and allowed the automatic downloading of new stories.

Nexus
A range of smartphones and tablets developed by Google. The Nexus range runs a pure version of Android.

NFC
Near Field Communication. A technology which allows data to be between phones or between your phone and another device.

Noise
The unwanted colour or luminance variations of pixels that degrade the overall quality of an image. Noise can result from several different sources including a low signal to noise ratio, the use of high ISO settings, long exposures, stuck sensor pixels and compression artefacts. It can appear as random colour speckles, a grain-like effect or banding.

Notification Centre:
A pull-down list of recent notifications, accessible from any iOS Home Screen or from within any iOS app. Similar to the Notification Panel found on Android.

O...

OEM
Original Equipment Manufacturer. A company which manufacturers devices for another brand (e.g. ASUS is the OEM of Google's Nexus 7.)

One Series
A range of smartphones from HTC. Includes the One X, One V and the One S.

Open GL
An open source graphics library, used on some smartphones.

Open Source
Software which is available to be studied, used and adapted by anyone. Android is open source software.

Operating System
Also OS. The program that's loaded into the computer after the initial boot sequence has completed. The OS manages all the other programs, graphical user interface (GUI), input and output and physical hardware interactions with the user.

Optimus
A range of smartphones from LG

OTA
Over The Air. A method which upgrades are wirelessly sent to smartphones.

Output
Data that is sent from the program to a screen, printer or other external peripheral.

P...

Pantech
A South Korean smartphone manufacturer.

PDF (Portable Document Format)
Developed by Adobe Systems, PDF is an open standard file format for cross-platform document exchange. PDF is highly extensible, preserves the integrity of the original document, is searchable and provides document security.

Photos
Built-in Apple app that handles your photo albums on your iPhone and iPod touch 4, and synced photos and videos for iPhone and all generations of iPad and iPod touch.

Photo Stream
Part of iCloud, Photo Stream stores your last thirty days or 1000 photos online and on your iOS devices, and all your photos on your Mac.

Pixel
Derived from the term picture element, this is the smallest unit of information in a digital image. It is also commonly used to describe the individual elements on a capture device such as a camera sensor.

PIN
Stands for Personal Identification Number. Used to lock smartphones and SIM cards.

Plug-In
A software application or module that provides extended and specific functionality from within a larger host application.

Portrait Mode
This describes a smartphone or tablet when you hold it vertically; this is when it's taller than it is wide and the Home button is at the top or bottom of the screen.

PSD
The .psd (Photoshop Document) format is a popular proprietary file format from Adobe Systems, Inc. It has support for most of the imaging options available in Photoshop, such as layer masks, transparency, text and alpha channels. In addition, spot colours, clipping paths and even duotone settings can be saved if you are preparing images for printing.

Project Butter
Software enhancements introduced in Android 4.1. Designed to smooth out frame rates and animations.

Q...

QR Code
A type of barcode which can be scanned by smartphones to reveal information such as text and website URL's.

QuickTime
Apple's 2D video and graphics player, used to play movies and other video on iOS.

R...

Raw Files
A Raw file is the unprocessed data captured by a digital camera sensor. In most cases, cameras write Raw files using a proprietary file format. Raw files give the photographer the advantage of managing image processing during post-production rather than letting the camera make the processing decisions, as happens when shooting in JPEG format. See also: DNG.

Recovery Mode
A separate operating mode of Android. Mainly used for device administration and repair.

Retina Display
Super sharp display available on Mac computers and iOS devices.

Resolution

A measurement of the ability of an optical, capture, or output system to record and reproduce detail. It can be defined in several different metrics such as Line Pairs, PPI, DPI, SPI and LPI. Also see DPI and PPI.

RGB
A colour model that uses the three primary additive colours (red, green, blue) that can be mixed in different ratios to make all other colours.

ROM
Read Only Memory. In Android a ROM is used to load software updates. Custom ROMs are software updates developed by third parties.

Root
In Android, to Root means to unlock the device to allow more access to the core software (or root).

Router
A device that manages and organises your home network devices, whether they connect to the router using a cable (LAN), or wirelessly (WLAN).

S...

Safari
Apple's web browser, both for Mac OS X and iOS (sometimes called Mobile Safari). Based on KHTML/ WebKit renderer and the Nitro JavaScript engine.

Samsung
A huge Korean smartphone and electronics manufacturer.

SD Card
A small memory card which can often be inserted into smartphones to increase storage capacity.

Sense
The user interface designed by and used on HTC phones.

Sharpening
The process of increasing or emphasising contrast around the edges of details in an image, to give the impression that the image is sharper than it really is.

Sideload
The process of installing an app onto your phone outside of the Google Play store.

SIM Card
The small plastic chip required in all GSM phones to connect to the mobile network.

Siri
Apple's intelligent virtual assistant, that replaces VoiceControl on the iPhone.

Sleep/Wake Button
Physical hardware button. Used to power on, wake from sleep, put to sleep and power down most smartphones and tablets.

Sony Ericsson
The company formed by Sony and Ericsson to manufacture and distribute mobile devices.

Sprint
One of the large US mobile carriers.

SSID
Service Set ID. In a nutshell, this is the 'name' of your wireless network, and can be changed using your router.

Super AMOLED
An improvement of AMOLED displays, providing brighter, less power hungry and less reflective screens.

T...

T-Mobile
Large US mobile carrier and manufacturer of smartphones.

Tegra 2
NVIDIA's dual-core mobile processor.

Tegra 3
NVIDIA's newer, quad-core, mobile processor.

Tethering
Using your smartphones data connection to provide internet access for another device (laptops, etc.)

Text Field
Any area where you can add text. For example, the search field is where you type something you're looking for. Tap on a text field to bring up the virtual keyboard.

Thumbnail Image
A small, low-resolution image preview used on the web to link to a high-resolution version of the file. Thumbnails can also be embedded in file formats such as TIFF and PSD.

TIFF or TIF (Tagged Image File Format)
An open standard file format specifically designed for images. TIFF can incorporate several types of compression, including LZW, JPEG and ZIP. The format is suitable for the storage of high quality archive images. The DNG format is based on the main TIFF standard.

TouchWiz
Samsung's custom user interface.

Twitter
One of the most popular social networks built around a follower and following system rather than friends.

U...

UPnP
Universal Plug and Play. A protocol used by digital media players for enjoying video, music, and pictures over your home network.

URL
Uniform Resource Locator. This is a web address, used to access a web page on the Internet, and usually starts 'www' and ends in '.com', or some other top-level domain.

USB
Universal Serial Bus. The connection method now used by most smartphones to connect to a computer or power source (MicroUSB).

V...

Vanilla
Sometimes used to describe Android without any custom user interface applied.

VDSL
Very High Speed Digital Subscriber Line. It's another protocol for getting on the Internet using your phone line, and is sometimes shortened to DSL.

Verizon
One of the four large US mobile carriers.

Virtual Environment
A cooperatively isolated runtime environment that allows Python users and applications to install and upgrade Python distribution packages without interfering with the behaviour of other Python applications running on the same system.

Virtual Machine
A computer defined entirely in software. Can be used to test/run/create code that won't affect the host system.

VPN
(Virtual Private Network):
This provides secure access over the Internet to private networks, such as the network at your company or school.

W...

While Loop
A coding loop that repeats code while a comparative statement returns the value True.

White Balance (WB)
In digital photography, white balance establishes the colour balance of the image in relationship to colour temperature of the lighting conditions. Most digital cameras have several built-in white balance presets (tungsten, daylight, cloudy, fluorescent, etc.) along with an auto setting and the ability to set a custom WB.

Widgets
The name given to the home-screen gadgets which allow you to see app updates, news, etc.

Wi-Fi
A group of backwards-compatible radio technologies used to connect peripherals to a network wirelessly.

WLAN
Wireless Local Area Network. Your network of wireless devices, as opposed to devices connected with a cable (see LAN).

World Phone
A device which works on both CDMA and GSM networks outside of its home country.

WPS
Wi-Fi Protected Setup, an easier way of connecting wireless devices to your router.

XYZ...

Xperia
A range of smartphones developed by Sony Ericsson. Includes the Xperia T and the Xperia Play, the PlayStation smartphone.

YouTube
Google-owned, web-based video streaming service. A YouTube app is usually pre-installed on Android devices.