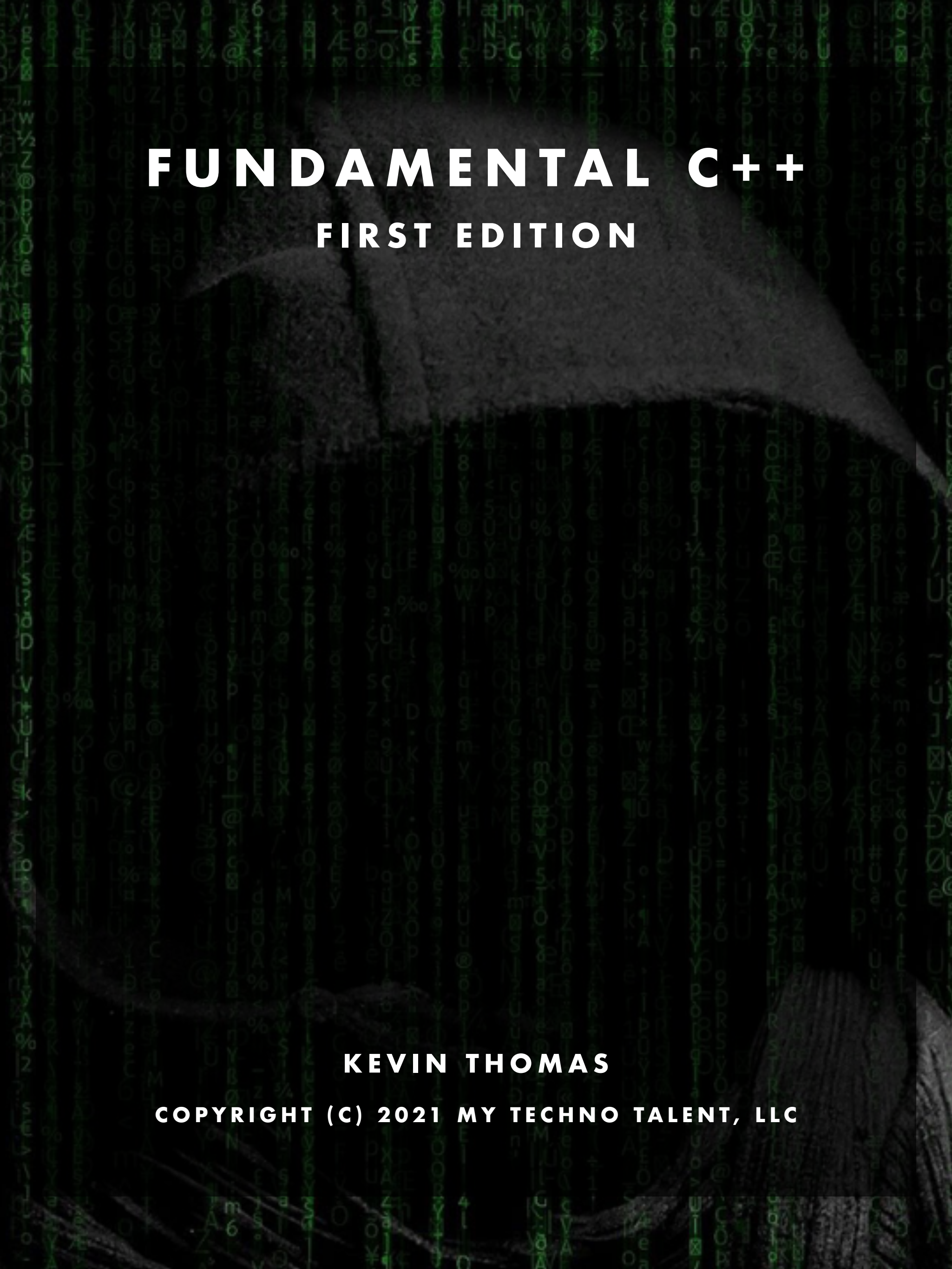

FUNDAMENTAL C++

FIRST EDITION

KEVIN THOMAS

Fundamental C++
First Edition

Kevin Thomas

FORWARD

C++ is one of the most powerful and scalable languages in existence today. C++ is used to create most of our modern web browsers in addition to powerful network security tools like Zeek. C++ also dominates within the gaming community.

We will begin our journey with the basics of C++ and then progress into variables and constants. We will then introduce arrays and modern C++ vectors and dive into the variety of C++ operators. At that point we will discuss proper input/output engineering and introduce the proper design and handling of program flow.

Once we have a grasp on those fundamentals we will introduce the concept of pointers and C++ smart pointers and work our way into functions and classes.

We will then cover inheritance, polymorphism and the STL or C++ standard template library.

GitHub repo @ https://github.com/mytechnotalent/Fundamental-CPP

CHAPTER1 - HELLO WORLD

As is tradition in any programming development we will begin with the famous hello world implementation and dive immediately into our first C++ application.

Let's begin by creating a folder for our course and our first project folder.

```
mkdir fundamental_c++
cd fundamental_c++
mkdir 0x0001_hello_world
cd 0x0001_hello_world
```

The first thing we want to do is create our Makefile. A Makefile allows us to properly build our applications in a scalable and proper fashion. We will call it **Makefile** and code it as follows.

```
main: main.cpp
    g++ -o main main.cpp -I.
    strip main

clean:
    rm main
```

Our first application will be **main.cpp** within the **0x0001_hello_world folder** as follows.

```
#include <iostream>

int main()
{
    std::cout << "Hello, World!" << std::endl;

    return 0;
}
```

Before we get started I want to share the C++ documentation to reference. It is good practice to review the docs as you begin architecting your designs.

https://en.cppreference.com/w/cpp

It is also important to keep up with the home of the C++ standard on the web.

https://isocpp.org

Let's begin by reviewing the *iostream* library.

https://en.cppreference.com/w/cpp/header/iostream

The iostream standard library header is part of the input/output library and handles basic IO within C++.

We use *iostream* so that we can utilize the libraries *cout* and *endl* objects. Let's start by looking at the *cout* docs.

https://en.cppreference.com/w/cpp/io/cout

The *cout* object controls output to a stream buffer which is STDOUT or standard output which in our case is our monitor.

The *endl* object or end line manipulator inserts a new line character into the output sequence and flushes the input buffer.

https://en.cppreference.com/w/cpp/io/manip/endl

Now that we have an understanding of what these two objects are we can review this code. We start with the *#include <iostream>* which the # is referred to as a preprocessor directive as we now have some familiarity with *iostream* from above.

We then see our main entry point or *main* function which is of an integer type. If everything in main completes successfully we return back an integer of *0* to the operating system.

We then see *std::cout* which std references the standard namespace. This design ensures that we do not have naming collisions. In a larger architecture we may have another *cout* which could be part of *foo::cout* such that if it was called there would be no naming collision with *std::cout*.

The *::* is called the scope resolution operator which denotes which namespace an object a part of in our case the standard name space or *std*.

We use the insertion operator << to insert, *"Hello, World",* which is what we call a string literal, into that stream and therefore send it to stdout, our monitor.

We then see another insertion operator << to insert a new line and flush of the input buffer.

The entire line starting with *std::cout* and ending with the semicolon, *;*, is what is referred to as a statement and all statements in C++ must end with a semicolon.

Finally we *return 0* to the operating system such that there was no error and end the program execution.

Let's now run **make** in our terminal.

We happily see, *Hello, World!*, echoed to our terminal.

Project 1

Your assignment will be to create a new folder **0x0001_about_me** and create a **main.cpp** and a **Makefile** and write a program that will tell us about yourself using what you learned above.