Cross platform C++ compiling for App development IOS and Android

Yes, it's **possible to create C++ modules for npm and use them in an Ionic-based mobile application**, but it involves a series of steps and technologies to bridge between JavaScript (which Ionic uses) and C++ code.

Here's an overview of how this can be achieved:

1. **Create a C++ Addon using N-API**: You can create a C++ addon by utilizing the Node.js N-API which allows you to write C++ code that can be called from JavaScript. These addons can then be compiled into binary format and published to npm.
2. **Use node-gyp**: node-gyp is a tool that enables you to compile your C++ code into a Node addon. You will need to write a binding.gyp file to configure the build process.
3. **Integrate with Ionic**: Once you have your C++ addon ready and published to npm, you can use it in your Ionic project by installing it like any other npm package. However, since Ionic is primarily intended for mobile development and uses a WebView, you will need a bridge to communicate between the WebView JavaScript context and the native code (C++ addon). You can use a plugin like cordova-plugin-native to achieve this.
4. **Platform Specific Configurations**: For iOS and Android, you may need to make some platform-specific configurations to ensure that your C++ code is properly compiled and linked for the respective platforms.
5. **Testing on Devices**: It is essential to test the app on real devices to ensure that the native code is executing as expected.

Here’s a simplified flow:

- Write your C++ code.
- Create a Node.js addon using N-API.
- Use node-gyp to compile the addon.
- Publish the addon to npm.
- Create an Ionic project.
- Install your addon from npm.
- Use a Cordova or Capacitor plugin to bridge between JavaScript and native code.
- Make any necessary platform-specific configurations.
- Build your Ionic app for iOS and Android.
- Test on real devices.

Please note that this is a high-level overview and each step can be complex, especially if you are not familiar with C++, Node.js addons, or Ionic. Be prepared to consult documentation and seek help from the community if needed.

Yes, you can use the same C++ code for an iOS app, but there are several steps and considerations involved in integrating C++ code into an iOS application. Here’s an outline of the steps:

1. **Writing Portable C++ Code**: Ensure that the C++ code you write is portable and does not rely on platform-specific features. Stick to standard C++ libraries that are supported across platforms.
2. **Create Objective-C++ Wrapper (Optional)**: iOS applications are primarily developed using Objective-C or Swift. If you need to interact with iOS-specific APIs, you might need to create an Objective-C++ wrapper. You can rename your files with the .mm extension to indicate that they are Objective-C++ files, which allows you to use C++ together with Objective-C.
3. **Add C++ Files to Xcode Project**: Add your C++ files to

your Xcode project. You can do this by dragging them into your Xcode project or by adding them through the File menu.

4. **Configuration in Xcode**: You will need to configure your Xcode project to use the correct C++ standard library (libc++ is common on iOS) and set the C++ language standard that your code requires (e.g., C++11, C++14).
5. **Linking Libraries**: If your C++ code depends on any libraries, you will need to link them in your Xcode project. This can be done under the “Build Phases” tab of your target settings.
6. **Write Interface Code**: Write the code that interfaces between your iOS application (Objective-C or Swift) and your C++ code. This usually involves writing functions that can be called from Objective-C or Swift, which in turn call your C++ functions.
7. **Testing on iOS Devices**: Since iOS simulators do not perfectly emulate the behavior of real devices, especially when it comes to native code execution, it is very important to test your application on actual iOS devices.
8. **Recompilation for iOS**: Yes, you will need to recompile your C++ code for the iOS platform. This is typically handled automatically by Xcode when you build your project.

Note: If you are using the same C++ code in an Ionic project as mentioned in your earlier question and also in a native iOS app, you’ll have two different environments (one is a hybrid mobile app, and the other is a native iOS app). The integration steps will vary for each, and you might need different sets of wrapper/interface code for each environment.