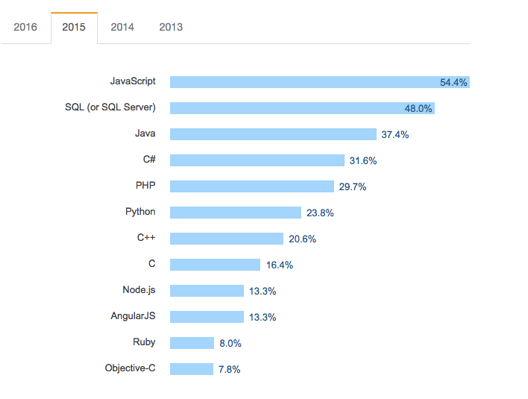
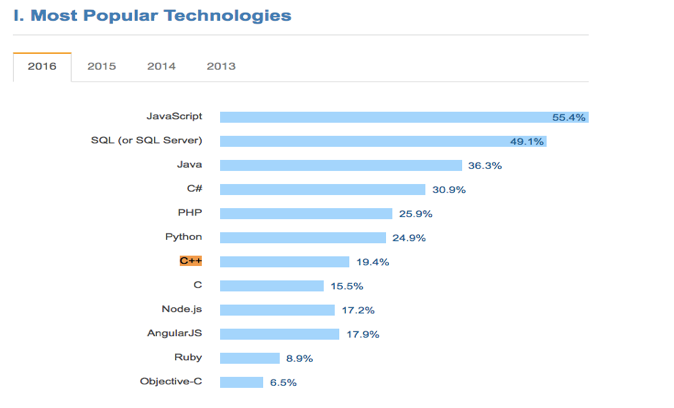
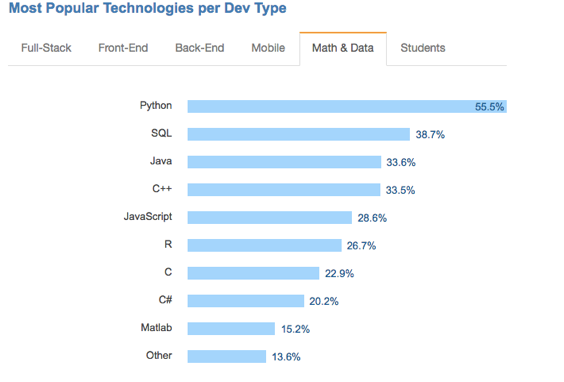
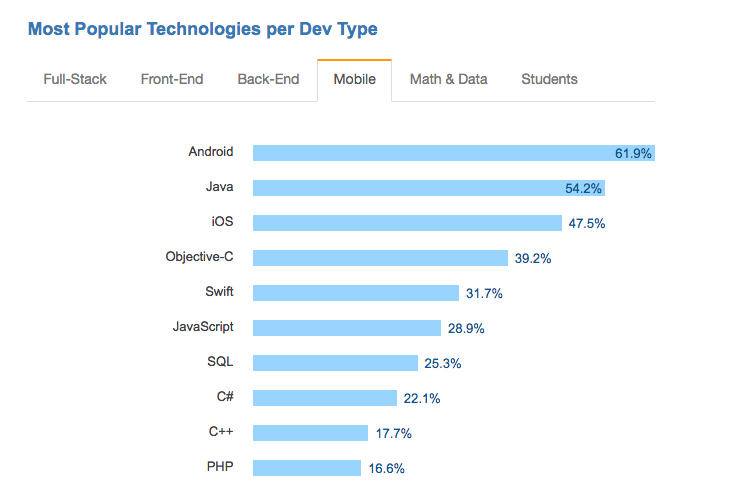
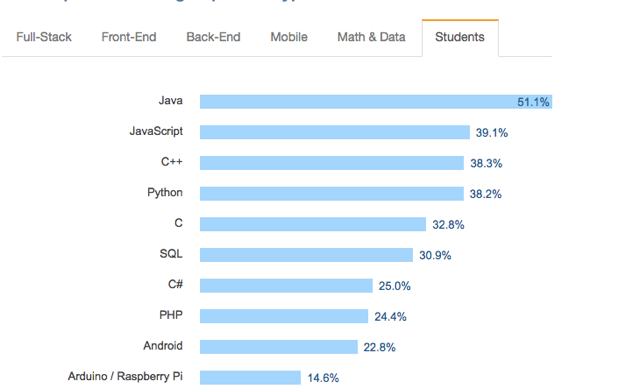
|  |  |  |
| --- | --- | --- |
|  | C | C++ |
| pros | You pick C when   * you need portable assembler (which is what C is, really) for whatever reason, * your platform doesn't provide C++ (a C compiler is much easier to implement), * you need to interact with other languages that can only interact with C (usually the lowest common denominator on any platform) and your code consists of little more than the interface, not making it worth to lay a C interface over C++ code, * you hack in an Open Source project (many of which, for [various reasons](http://programmers.stackexchange.com/questions/103897/is-the-c-programming-language-still-used/103904#103904), stick to C), * you don't know C++.   In all other cases you should pick C++. | Pretty fast comparing with java, python etc.  OOP(so you can have class)  Has Boolean type  Most importantly, easy to find a tutorial online.  Reasonably written C++ will generally be at least as fast as C, and often at least a little faster. (but looks like our small program won’t be slow on a modern computer anyway. But it’s good to be fast?)  \*Reduced maintenance effort  int \*x = new int;  int \*x\_array = new int[10];  delete x;  delete[] x\_array;  C++ has a much larger library  int main()  {  printf( "Hello, World" );  }  note that you don’t need to return 0; it’ll be done automatically. |
| Cons: | C is not OOP.(structure)  C 89 doesn’t have boolen type  int \*x = malloc( sizeof(int) );  int \*x\_array = malloc( sizeof(int) \* 10 );  free( x );  free( x\_array );  Very little safety net. | It is not very safe by itself, as it lacks automatic boundary checks, invalid pointer checks etc.  By default there is no built-in memory management, requiring developers to use external libraries or re-invent the wheel.  Has a somewhat more difficult learning curve than some other languages. For instance C++'s most powerful features, such as templates, have a very complex syntax.  GUI, Networking and Threads aren't standardized, requiring the use of non-standard, third-party libraries |

Statistical aspect for language selection.



**For android apps:**

According to “developer.android.com”

The Native Development Kit (NDK) is a set of tools that allows you to use C and C++ code with Android.

But the NDK may not be appropriate for most novice Android programmers who need to use only Java code and framework APIs to develop their apps. However, the NDK can be useful for cases in which you need to do one or more of the following:

Squeeze extra performance out of a device to achieve low latency or run computationally intensive applications, such as games or physics simulations.

Reuse your own or other developers' C or C++ libraries.

.

**For IOS apps:**

<http://stackoverflow.com/questions/10289890/how-to-write-ios-app-purely-in-c>

“you can write in c, but it’s not practical”

<https://www.quora.com/Can-I-use-C++-in-Xcode-for-iOS-development>

<http://philjordan.eu/article/strategies-for-using-c++-in-objective-c-projects>

For OS X:

http://stackoverflow.com/questions/5915656/jumping-from-ios-to-osx

http://stackoverflow.com/questions/6553305/difference-between-iphone-development-and-mac-osx-development

simple conclusion: core code (C++ code we’ll be writing in this case) would be the same with IOS. Interface would be different.

For Windows 7/8/10 :

<https://msdn.microsoft.com/en-us/library/jj153215.aspx>

<https://msdn.microsoft.com/en-us/library/dd831853.aspx>

<https://www.quora.com/What-is-the-programming-language-used-to-develop-Windows-10>

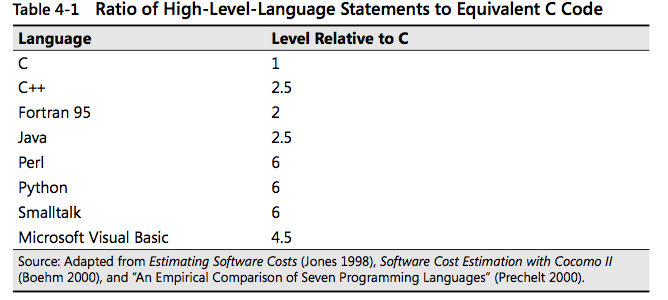
C and C++ are both good options.

Some people suggest using a combination of C and C++

***About portable C/C++ code.***

[***http://stackoverflow.com/questions/30056072/porting-msvc-code-to-android-ios***](http://stackoverflow.com/questions/30056072/porting-msvc-code-to-android-ios)

***code complete 2nd edition:***

******

**C**

C is a general-purpose, mid-level language that was originally associated with the UNIX operating system. C has some high-level language features, such as structured data, structured control flow, machine independence, and a rich set of operators. It has also been called a “portable assembly language” because it makes extensive use of pointers and addresses, has some low-level constructs such as bit manipulation, and is weakly typed.

C was developed in the 1970s at Bell Labs. It was originally designed for and used on the DEC PDP-11—whose operating system, C compiler, and UNIX application pro- grams were all written in C. In 1988, an ANSI standard was issued to codify C, which was revised in 1999. C was the de facto standard for microcomputer and workstation programming in the 1980s and 1990s.

**C++**

C++, an object-oriented language founded on C, was developed at Bell Laboratories in the 1980s. In addition to being compatible with C, C++ provides classes, polymor- phism, exception handling, templates, and it provides more robust type checking than C does. It also provides an extensive and powerful standard library.

***From Code Complete 2nd edition, Microsoft press***