ECM2433 – The C Family

Continuous Assessment 2

The History and Applications of C, Objective C, C++, and C#

660025581

The following report is a brief critical analysis of the programming languages C, C++, C#, and Objective C. This report will discuss their shared history and their relationships with modern languages, a comparison of the key aspects of the languages (similarities and differences), and an illustration of the application areas to which each language is best suited.

Word Count: 1642

I certify that all material in this dissertation which is not my own work has been identified

History and Relationships

In 1971, Dennis Ritchie and Ken Thompson created the C programming language, originally designing it for and implementing it on the UNIX operating system on a PDP-11 minicomputer. It was developed from the B programming language, itself a simplified version of BCPL, also written by Ken Thompson, and in doing so, changed it from an interpreted language to a compiled language, introducing the concept of types such as characters and integers. [1]

In 1983, the American National Standards Institute (ANSI) established "an unambiguous and machine-independent definition of the language C" [1], known as ANSI C, which would later be adopted and modified by the International Organization for Standardization (ISO).

In 1979, Bjarne Stroustrup began developing "C with Classes", a language designed to implement the facilities available in Simula, one of the first object-oriented programming languages, "with C's efficiency and flexibility for systems programming" [2]. Due to its success, Stroustrup designed C++ in 1983, "a cleaned-up and extended successor to C with Classes" [2], which was viewed as a separate language from C.

In the 1980s, Brad Cox and Tom Love began development of a different Object-oriented extension of C adding some of the abilities of another object-oriented language, Smalltalk. This would later be marketed as Objective-C through Cox and Love's own software company Productivity Products International/StepStone. The rights to use Objective-C was licensed by NeXT, in 1988, before being later acquired by Apple when it bought NeXT in 1996 [3]. Objective-C would go on to influence Apples own developed language Swift, referring to it as "Objective-C without the C" [4].

In early 1999, as part of the development of Microsofts .NET framework, a team led by Anders Hejsberg began development on a new language called 'C-like Object-Oriented Language' (COOL). However, by 2000 when .NET was publicly announced the language had been renamed to C# for trademark reasons [5]. There was also some controversy surrounding the release of C# over the similarities between itself and Java, which was released a few years prior.

Many other languages created have drawn inspiration from C and C++, ranging from its syntax to the way in which the objects communicate. Languages such as Python [6] took inspiration from the extensive libraries and extensions that C provided/ provided the ability to create, implementing similar features into Python.

Comparison

Paradigms

Of the languages covered, C is the only one that does not follow an Object-Oriented paradigm, instead following a procedural paradigm. This means that a C program is only concerned with the computational steps required to perform some task. Conversely, Objective-C is the only language covered that **only** follows an object-oriented paradigm, as both C# and C++ are both procedural and object-oriented. In addition, C# and C++ can be used, in limited capacity, as a functional programming language due to features implemented in later releases [7] [8].

Of those languages that implement an object-oriented paradigm, the way in which the objects interact differs slightly. Both C++ and C# call methods on objects via the more common approach where the method name is bound to a section of code in the class by the **compiler**. Objective-C, however, implements a model of object-oriented programming that instead passes messages between instances of objects, the target of which is resolved at **runtime**. Once an instance receives a message, it will then itself interpret it and respond accordingly, which could include forwarding the message to another instance that can respond to it or not respond at all. Therefore, there does not exists any type checking as an instance may never respond [3].

Further, the way in which inheritance works in these object-oriented languages also differs. C# does not allow for one class to directly inherit from more than one parent class to simplify the architectural structure. However, C# does allow for a class to inherit any number of interfaces, allowing for multiple inheritance to be replicated [5].
Objective-C also disallows multiple inheritance, however its behaviour can be replicated using protocols which can either be formal, where all defined methods must be implemented, and informal, where any number of the methods can be implemented. Further, one may use categories to add further functionality to a class at runtime without the need to recompile [3].
On the other hand, C++ allows multiple inheritance, where a class can inherit from multiple base classes. Further, each inheritance can be allocated an access modifier which specifies whether derived and unrelated classes can access the public and protected methods of the base class. Interfaces can also be used within C++ but are better known as abstract base classes and are implemented using only virtual functions [9].

Typing

All four of the discussed languages allow for static typing, that is that the type checking occurs during compilation time. Of those, C, C++, and C# can **only** be statically typed meaning that the data type of each variable must be declared before it is instantiated.
Objective-C can use static typing by assigning further information to variables to give the compiler more information. However, out of the discussed languages, it the only one that can use dynamic typing, meaning that the type checking occurs at runtime. This is partially due to the message passing that is used within the language, as an instance can be passed a message that it cannot understand due to it not being defined within the class [3].
Further, it can be argued that Objective-C is the only language of the discussed that is strongly typed as, unlike the others, you cannot freely cast one type to another. Of the remaining three, C# would be considered the strongest of them, despite still being weakly typed, as there exist more restrictions on what can be cast.

Memory Access

The PDP-11 minicomputer that C was first run on had a very small amount of memory, meaning that the language needed to use its memory resources efficiently. As a result, C allows for memory to be explicitly allocated, reallocated, and for this memory to be 'freed' once it is no longer required [1].
C++ also allows for this explicit memory management through the same keywords although this is discouraged as it does not initialise or destroy objects created. As such, the keywords **new** and **delete** are used in place which remove the need for manual memory allocation and freeing whilst also initialising and destroying object instances [9]. However, due to both C and C++ requiring that allocated memory must be freed at a later point, there exists the problem of **memory leaks** where memory is never released and cannot be accessed any more. This can result in memory issues occurring where a program may not have enough memory to store data.
C# also allows for explicit memory access; however, it is not encouraged as this can be managed by the common language runtime in the .NET framework [10]. For this reason, all memory address pointer can only be used within block of code that are marked as **unsafe**. This is because C# only accesses objects via safe references, which are either NULL or point to an object that is currently in use. Further, C# introduces **garbage collection**, meaning that unused/out of scope references are automatically freed preventing any memory leaks [5].
Objective-C manages memory through a process called reference counting, which allows for a single object to have many owners and, so long as the object retains at least one owner, it will "stay alive".

Once an object has no owners it is deallocated from memory. From 2011, the memory allocation and deallocation calls are inserted into the code at compile time to reduce development time, and although the manual memory allocation methods still exist, they are discouraged [11].

Application areas

One of the main benefits of using C is its speed and memory efficiency and, for this reason, it is commonly used in operating systems and embedded systems. C was initially created for scripting a UNIX operating system and has since become an integral part of the development of other operating systems such as Linux-Kernel, which is a large part of Android [12].

Games development is an industry that heavily uses C++ as a main language. This is, again, due to its speed and memory efficiency but also because it has a greater control of the hardware of the machine. Further, the object-oriented paradigm lends itself very well to the concept of a video game, where "real-world" objects and interactions need to be modelled. Large games engines such as Source (Valve) [13] and the Unreal Engine [14] both use C++ as a primary language.

The main area of use for Objective-C is within Apple products and services since Apple own an exclusive license for the language. Objective-C's dynamic binding is beneficial to this area as many Apple products contain a well-designed and highly interactive user interface which can all be resolved at runtime. This also allows "developers [a] freedom of expression in their design" [15]. Further, this also allows for information about running applications to be retrieved and thus be able to create further tools that can monitor the activities of other Objective-C applications [15].

The main benefit of using C# is its inclusion in the .NET framework and thus, similarly, its main area of use is for the development of Microsoft and Windows products and services. The .NET framework provides developers an extensive class library to support code-reuse for "common low-level programming operations" and automatically manages memory usage in the common language runtime [16].

Conclusion

In conclusion, the C programming language is one of the pioneers of modern day computer science, if not one of the most pivotal languages of all time. It has set the standard for how high-level languages are to be written and has inspired many more languages, some directly related to it and some not. Of the four languages discussed in this report, it is not fair to say that one language is better than all others as they all have areas in which they are better suited for.

References

[1] B. W. Kernighan and D. M. Ritchie, The C Programming Language 2nd Edition, New Jersey: Prentice Hall , 1988.

[2] B. Stroustrup, “A History of C++: 1979-1991,” in *History of Programming Languages*, Cambridge, Massachusetts, 1993.

[3] S. G. Kochan, Programming in Objective-C 3rd Edition, Addison-Wesley, 2011.

[4] Youtube, “Apple - WWDC 2014,” Apple, 3 June 2014. [Online]. Available: https://youtu.be/w87fOAG8fjk?t=1h44m19s. [Accessed 14 March 2018].

[5] Ecma International, C# Language Specification 5th Edition, Geneva: Ecma International, 2017.

[6] B. Venners, “The Making of Python,” Artima, 2003. [Online]. Available: https://www.artima.com/intv/pythonP.html. [Accessed 14 March 2018].

[7] B. McNamara and Y. Smaragdakis, “Functional Programming in C++,” Atlanta, 2000.

[8] E. Buonanno, Functional Programming in C#, Manning Publications, 2017.

[9] B. Stroustrup, The C++ Programming Language 4th Edition, Addison-Wesley, 2013.

[10] Microsoft, “GarbageCollection,” 2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/garbage-collection/index. [Accessed 14 March 2018].

[11] Apple, “Memory Management,” Apple, 2012. [Online]. Available: https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/MemoryMgmt/Articles/MemoryMgmt.html. [Accessed 14 March 2018].

[12] Android, “Andoid Architecture - Kernel,” Google, 2017. [Online]. Available: https://source.android.com/devices/architecture/kernel/. [Accessed 14 March 2018].

[13] Valve, “Source Engine Features,” Valve, [Online]. Available: https://developer.valvesoftware.com/wiki/Source_Engine_Features#Programming. [Accessed 14 March 2018].

[14] Unreal Engine, “Introduction to C++ Programming in UE4,” Epic Games, [Online]. Available: https://docs.unrealengine.com/en-us/Programming/Introduction. [Accessed 14 March 2018].

[15] Apple, “Why Objective-C?,” Apple, 2010. [Online]. Available: https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/OOP_ObjC/Articles/ooWhy.html. [Accessed 14 March 2018].

[16] Microsoft, “Get started with the .NET Framework,” Microsoft, 2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/get-started/index. [Accessed 14 March 2018].