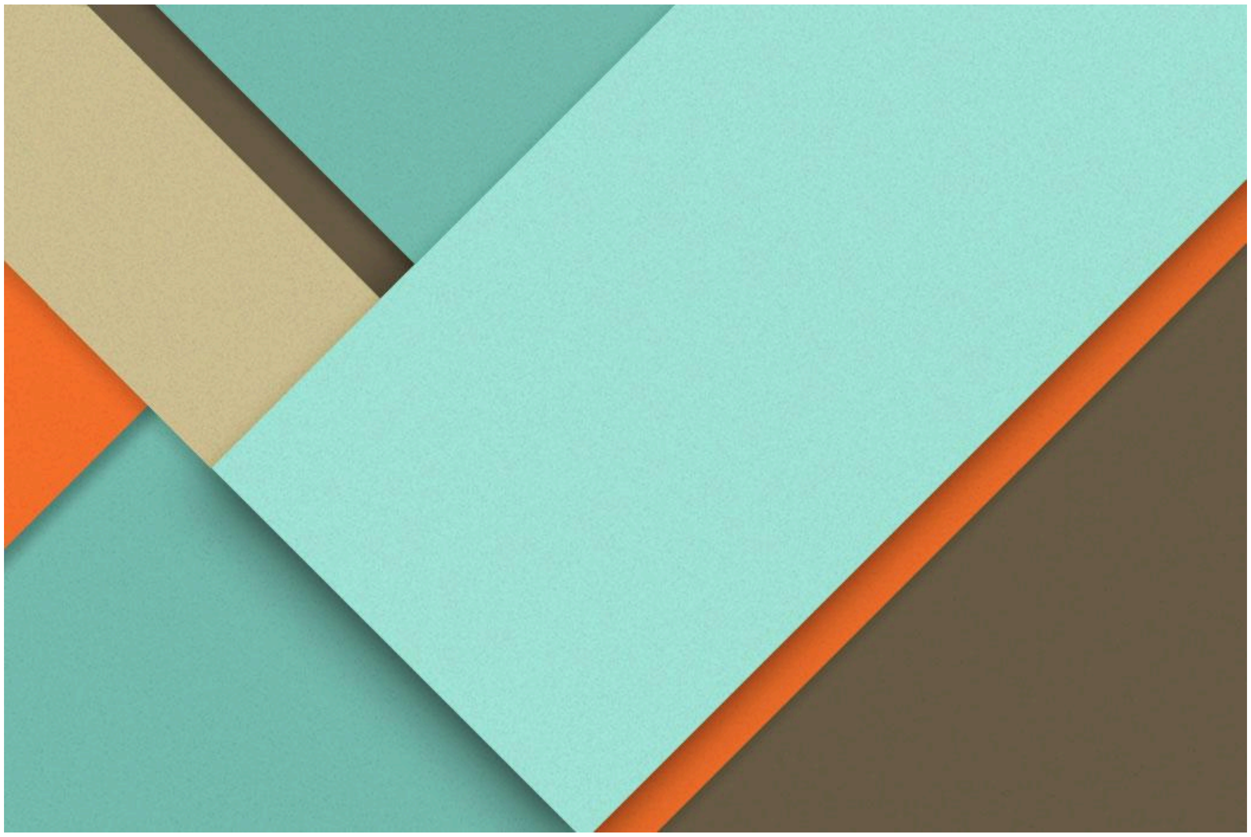

Object Oriented Languages Being Popular

04.03.2024

Samir Khadka
San Francisco Bay University
CS360 - Programming in C and C++

Abstract

Object-oriented programming (OOP) has changed the way we build applications and C++ is a well-established high-level programming language. In spite of the increase of new languages, C++ is still very much relevant and in demand thanks to its strong features and diverse applications. This paper examines the structural specifics of C++ as an OO language versus those of Java and Python, and demonstrates practical instances where C++ performs better. This paper offers a comprehensive review of academic journals, industry reports, and online resources on C++ which focuses on the object-oriented programming paradigms supported by it, unique features it has compared to Java and Python, and most commonly used applications in operating systems, game development, embedded systems, and financial software.

C++ is a rich language with built-in features which enhance programming such as classes, objects, inheritance, polymorphism, encapsulation, and abstraction. Unlike Java and Python, C++ gives developers more authority over memory management using tools like pointers and manual memory allocation. The provision of the function to memory management is what makes C++ one of the most suitable languages for applications that require high performance and accuracy of resource use, such as operating systems development and real-time embedded systems.

Unlike Java, which is platform independent thanks to its bytecode compilation and virtual machine execution model, and Python, which values simplicity and readability, C++ is good at performance and flexibility at the same time. The equilibrium of C++ allows it to thrive in fields like game development where high performance and low-level capabilities are essential in order to build highly immersive gaming experiences. Additionally, C++ is commonly used in financial software development, where its speed and low latency are vital for implementing high-frequency trading algorithms and other complex financial models. The paper has shown C++ to be as relevant in modern software development as it was when it was designed and also that it can be used across multiple domains of applications.

Introduction

Object-oriented programming (OOP) has revolutionized the world of computer programming providing a mental model where modules, modularity, are of the paramount importance, aren't they? However, among the many omnipresent programming languages that follow some of the OOP principles, C++ is still the only language that is greatly known for its robustness, high performance efficiency and an adaptability to a wide range of problems. Generally, C++, the second-generation language developed by Bjarne Stroustrup since the early 80s, is an integral part of the assortment of high-level programming languages that have dominated the niche for decades, rendering its capacity of service across a gamut of fields, from the system level programming to advanced software development projects.

In this write-up, we investigate the persistent significance of C++ in the context of object-oriented programming languages such as Java and Python, unveiling its structural details, and also recognizing the prosperity it has brought for software development. C++ has a lot of methodology that gives you the ability to use object-oriented design patterns, among these are classes, objects, inheritance, polymorphism, encapsulation, and abstraction. These elements allow developers to code in a modular, maintainable way with appreciable extension and reuse capabilities, hence increasing efficiency while reducing time to market for software projects.

Along with the new languages and the frameworks, the C++ language is estimated to be the most popular for both system software and application development. It is ideal for projects achieving outstanding performance, doing its best in the low-level capabilities as well as with the efficient resource management, hence, it is these projects where speed, control, and scalability are essential. Furthermore, C++ combines performance and flexibility in an ideal sense, allowing programmers to choose the distinction between high-level abstractions and low-level optimizations according to the project conditions. This study entails a comparison of c++ features, applications and its comparative advantages for them to know reasons why they are still widely applied in industry sectors all over.

Methodology

The methodology followed in this paper utilizes the broad search of academic sources, industry reports, and online documents in order to collect the necessary information about C++, Java, and Python. This literature review includes scientific articles, books, official documentation on language, as well as related forums of developers and programmers. We plan to process information from different sources to deliver a comprehensive appreciation of the structural details, distinctive features, and application of C++ together with other object oriented languages.

Moreover, we use a variety of sources such as industry cases studies, software archives, and development forums for providing examples of current popular C++ designs, systems, and platforms. The proposed course will use practical examples and case studies to demonstrate how C++ is popular and applicable in a number of areas such as the operating systems, gaming, embedded systems and financial software. These indicative examples help flesh out the theoretical information provided in the literature review and emphasize the important role of C++ in today's software development. Through this approach, we aim to give the most complete presentation of C++ as an object-oriented language and highlight its ever-existing usefulness in modern software engineering.

Discussion

I. C++ OO Language Structures:

C++ is well known in programming languages owing to the fact that it offers much better support for object-oriented programming (OOP) methods than other languages. Fundamentally, C++ as a language ends up being enabled with numerous in-built capabilities that facilitate the adoption of OOP paradigms. These are classes which actually are blueprints representations made in order to create objects, and objects that hold data and behaviors. Inheritance provides classes with the ability to incorporate properties and methods from the ancestral class. Thus, inheritance facilitates code reuse in nested structures. Thanks to the polymorphism classes are able to encapsulate distinct algorithms according to the object types, which improves the application's extensibility and flexibility. Data authentication and abstraction by the application of encapsulation — a key concept of OOP (Object-Oriented Programming) — includes the restriction of access to the objects' internal (hidden) components. Abstracted procedure in addition, allows engineers to spotlight essential aspects of systems and conceal intricate details of its working. Respectively, this will optimize developers to write clear, maintainable code, which then ends up to be flexible so as to adapt to changing requirements, thus resulting in high productivity and the quality of the software.

Differentiating features from languages like Java and Python, C++ grants you full access over memory management, enabling more detailed control. Java/ Python uses the garbage collection method for automatic memory management whereas C++ users have the choice of going into memory management manually through features like pointer and manual memory allocation. The memory management which is the process that brings challenges like memory leak and dangling pointers yet at the same time is a process which is the basis of C++ reason why it is the best choice when the parameter is about the optimization of the performance. This fine-grain handling of memory allocation allows the programmers to make a custom fit of their code onto different hardware architectures leading to optimization of performance for situations where resource not abundance matters, such as in embedded systems and high-performance computing.

Furthermore, the coding language, C++, is also capable of supporting both the procedural and object-oriented programming paradigms to an extent that it becomes the most suitable for different development contexts. While object-oriented programming encourages modularity, reusability, and abstraction of data, procedural programming offers implementations of algorithms and data structures. The smooth ability to switch between programming styles helps developers to use the qualities of each paradigm and to respond to project's requirements. Therefore, C++ is used for various areas of expertise, whether system software, software development, game development, scientific computing and many other things. Because of this adaptability, C++ is considered the preferred option for projects that require either performance or flexibility and strengthened the importance factor more than ever for the software development foundation.

II. Comparison with Java and Python:

Java and Python, along with C++, are among the most popular OOP languages with their unique functional definitions and strong points. Java, which has traditionally offered platform independence, achieves this by means of bytecode compilation and execution on a virtual machine (JVM). This model enables the 'Write once, run anywhere' paradigm since the Java programs can be executed on any given platform as long as it has a compatible JVM and thus is preferably used on cross-platform developments. While run-time environment portability is emphasized in Java, control and performance are the two pillars of C++. As opposed to JVM's bytecode execution model, direct compilation to machine code in C++ offers simplicity of design as well providing the capability to access underlying systems directly resulting in reduced overhead. Hence Java is widely used when platform independence is vital – for instance to create a web application or an enterprise system – but C++ is excellent in dealing with performance optimization in cases where it is crucial – such as systems software and game development.

On the downside, though, Python is far from the other two languages – C++ and Java – in the issues of simplicity, readability, and flexibility. That is something to pay attention to. This is why the fact of Python's concise syntax and dynamic typing is a sure pass for the beginner-friendly nature and high-pace capabilities. On the contrary, Python adopts an automatic memory management through garbage collection, thereby lightening the work of the developers from being doing dull tasks like manual memory allocation and deallocation. Hence, Python offers developers an easier way of coding and minimizes the possibilities of memory-related mistakes. This is one reason why it is widely used in projects that emphasize development speed and simplicity rather than performance optimization. This gives rise to the fact that Python is popular in various sectors including web development, data science, artificial intelligence and scripting wherein its advantages of being versatile and crisp are evident.

While these three languages may vary in many ways, they all, however, maintain some common themes in feature support for object-oriented programming principles. Through these tools developers are able to design codebases that are modular, maintainable and can introduce classes, objects, inheritance, abstraction, polymorphism, encapsulation, and composition. Nevertheless, in the case of memory management, portability, and syntactic constructs, C++, Java, and Python differ in their degree of control, which make them best suited for different application areas. This awareness can guide developers on picking a priori the best language of their choice for their project based on both performance demands and constraints as well as project targets.

III. Examples of Current Popular Designs, Systems, and Platforms in C++:

Operating Systems:

C++ is inherent to the process of making Operating Systems mainly due to its performance and efficiency characteristics. As the operating systems are the central software that controls as well as enables computer hardware to work properly and create an environment for application software to run smoothly, there is a need for a programming language to handle low-level tasks properly. C++ can fill the shoes of this prerequisite very well because of its ability to directly interact with hardware and manage system resources in the most effective way possible. As a consequence, two popular operating systems, Windows, MacOS, and a few distributions of Linux, are generally mostly coded in C++. This type of system uses the C++ language as a platform for obtaining maximum performance, dependability, and adaptability to various hardware configurations and environments.

In addition to its ability to find this delicate balance in the performance and maintainability, C++ is one of the main factors driving its wide adoption in operating system development. Normally Operating systems need to have stable and stable code that is both efficient and able to confront various conditions without any crashes. One of the strengths of C++ is its support for low-level programming, including direct memory access and hardware manipulation. This translates into increased ability of developers to optimize crucial parts of an operating system in terms of efficiency and reactivity. Lastly, C++'s abundant library system and frameworks add to its flexibility. This allows developers to execute advanced tasks and fulfill the need of highly sophisticated dependencies that nowadays modern operating systems face.

Game Development:

The C++ remains the top language of game development, factually caused by low-level capabilities of processing and exceptional performance. The game development more frequently needs to exercise close control over hardware resources with the requirement for superior performance while providing elite gaming experience. With the capacity to directly address hardware and shared memory resources, C++ puts in the developer's hands all the tools necessary to optimize system performance for high-load tasks specific to game development. This development leads to the fact that C++ continues to be the language that is commonly used while the game engines are being developed and the complicated logic is implemented.

Let examples of common game engines such as Unreal Engine and Unity be the proof of C++ significance in the game industry. Among these game engines, used by the developers community to realize different types of games for the many platforms around the globe, C++ provides the core of the implementation. Apart from that, C++ assures game engine designers to make rendering pipelines, physics simulations, audio processing and other performance-hungry modules with their utmost. Another feature of C++ is his ability to work integrated with scripting languages such as UnrealScript or C# which offers developers the possibility to use C++ performance in critical tasks and the benefits of higher-level languages including flexibility and rapid prototyping capabilities. At the end of the day C++ ability to outperform in game development clearly shows its special capacity to deliver top gaming experience.

Embedded Systems:

C++ is key to the creation of embedded systems where these parameters, resource efficiency and timely performance are crucial. Entering systems embedded, covering equipment from autos to electronics of devices of Internet of Things, it is necessary to create software solutions capable of working within the strict streamlines of resources and guarantee reliability. C++'s balancing act between being high-level and relatively giving low-level control makes it a tool of choice for systems that require software development. Through its managed memory and code optimization, developers can design embedded systems' applications which are in limited space but powerful.

C++ is heavily used in car systems for the production of software components that are the creators of car features such as engines, infotainment, and advanced driver's assistance systems (ADAS). The quick reaction and reliability that C++ gives are fundamental to the safe and dependable automotive systems guarding of. Correspondingly, in cases where programming like consumer electronics and IoT devices, C++ allows developers to code firmware and applications in such a way that efficiently uses hardware resources and meets the requirements of low power operation and connectivity. It may well be that it is necessary to manage sensors in smart home devices, or communication protocols used in the IoT devices. There is no doubt that C++ gives you the flexibility and efficiency required in development of embedded systems for a wide variety of application domains.

Financial Software:

C++ is widely used for the development of financial programs, and its application domains range from HFT systems to algorithmic trading platforms. The finance sector is the one which compelled their software solutions to execute the strategies that are highly complex with speed, accuracy and low latency. C++'s performance-driven capabilities, which include optimized memory management and direct hardware access, make this language one of the preferred choices for building trading systems that can process large amounts of trading data and execute trades in real time. High frequency trading that involves milliseconds and which determines profitability in the financial sector highly relies on C++ for two purposes: first, to optimize code for speed and subsequently, to minimize latency.

Automated trading platforms, which are based on programming C++ to perform trading strategies, are another major user of C++ for their development. These channels need software which can handle market data in the current state, analyze complicated trading algorithms and execute orders without any lag. C++ low-level capabilities and high performance allow developers to generate sophisticated trading algorithms in a time-efficient manner. In addition, the compatibility of C++ with existing financial libraries and frameworks gives the language an added advantage in that the developers can leverage robust and scalable resources in order to build powerful trading systems. In general, C++ speed, low latency, and flexibility are the reasons for the choice of this language in creating financial software that operates in fast markets and in tough competition.

Result:

The results show that C++ keeps its relevance and is widely used by many areas of software development – particularly critical. Crystalized in the above scenario, C++ is the favorite to be used in projects in which performance, efficiency, and control over system sources are the most important. The language supports object-oriented programming concepts as well as provide the low-level capabilities which imagine the developers to create robust and scalable software implementation across an array of applications.

Moreover, C++'s prominence in the development of operating systems, game development, embedded systems, and financial softwares suggests that its performance is highly dependent on different use cases. No matter whether it's powering the core functions of Windows and macOS OS, offering an experience of immersion in the most popular game engines like Unreal Engine and Unity, or enabling immediate response in engines and financial platforms for trading, C++ forever proves its versatility and indispensability with the development of the more and more complex software. Ultimately, the outcome illustrates the continuing relevance of C++ and puts the technology in a leading position among software development techniques currently practiced.

Conclusion:

The summary shows that the analysis conducted in this paper demonstrates the unwavering importance of C++ among the elite languages founded on the high level of processing. The recent appearance of other languages and frameworks notwithstanding, C++decenyas its importance and popularity as a result of its brilliant features, diversity, and robustness. C++ combines the latest features of object-oriented programming paradigms with the essentials of low-level programming. As a result, it is a powerful tool for efficient, scalable, and maintainable development projects across all fields of application.

Along the way, the discussion of differences from other object-oriented languages, such as Java and Python, is done. C++ is shown as a more powerful and efficient language as it provides a greater control over memory management, performance optimization, and flexibility. Whilst Java gives importance to execution independence and Python considers usability and readability, C++ is ideal for applications which will run fast, are useful and give an individual over the project because it combines these two. On top of that, real use situations in operating systems, game making, embedded systems, and financial software development revealed broad applicability of C++ to even the most unrelated domains.

After concluding this paper, I can confirm the persisting credibility of C++ and its key role as a primary technology in modern developed software practices. The C++ programming language is still solidly here, despite the fact that technology, as it keeps accelerating, remains indispensable, driving innovation and being a backbone of systems and apps in all kinds of industries. Through mastering the fundamentals of C++'s structure, exploring the strengths and weaknesses of C++, and being aware of practical deployments, the developers can make a rational decision to choose the right language for their projects since the successful project in the digital environment is attributed to proper software solutions.

References

C++ vs python - what you need to know. KO2 Recruitment. (2023, August 23). https://www.ko2.co.uk/c-plus-plus-vs-python/#:~:text=C%2B%2B%20is%20commonly%20used%20in,analysis%20and%20backend%20web%20development.

GfG. (2023, October 6). *Object Oriented Programming in C++*. GeeksforGeeks. https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/

Jorge, J. (2019, November 25). *An overview of build systems (mostly for C++ projects)*. Medium. https://julienjorge.medium.com/an-overview-of-build-systems-mostly-for-c-projects-ac9931494444

(PDF) why C++ is not just an object-oriented programming language. (n.d.-a). https://www.researchgate.net/publication/2336195_Why_C_is_not_just_an_Object-Oriented_Programming_Language

S, R. A. (2023, February 21). *C++ VS Python: Overview, uses & key differences: Simplilearn*. Simplilearn.com. https://www.simplilearn.com/tutorials/cpp-tutorial/cpp-vs-python