

ТЕОРЕТИЧЕСКАЯ ИНФОРМАТИКА, КИБЕРНЕТИКА/THEORETICAL INFORMATICS, CYBERNETICS

DOI: https://doi.org/10.60797/COMP.2026.10.2 EDN: WABDLS

СРАВНИТЕЛЬНЫЙ АНАЛИЗ НАИБОЛЕЕ ПОПУЛЯРНЫХ СИ-ПОДОБНЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Научная статья

Большаков К.П.[1], Халиман В.В.[2,] *
[2] ORCID : 0009-0007-4657-9440;
[1, 2] Дальневосточный государственный университет путей сообщения, Хабаровск, Российская Федерация

* Корреспондирующий автор (vv.khaliman[at]mail.ru)

Аннотация

В работе приведён детальный сравнительный анализ наиболее популярных языков программирования, унаследовавших С-подобный синтаксис: Rust, Go, C# и Java. Рассмотрены ключевые характеристики, а именно производительность и эффективность использования ресурсов. Анализировались поддерживаемые парадигмы программирования и подход к системе типов. Также изучены принципы управления памятью и уровень обеспечиваемой безопасности (память, многопоточность). Кроме того, оценивалась простота изучения и богатство экосистемы.

На основе сопоставления сильных и слабых сторон каждого языка для различных задач был сделан аргументированный выбор с оптимальным балансом характеристик, наиболее соответствующим заданным техническим требованиям и целям проекта.

Ключевые слова: язык программирования, СИ-подобные языки, C, C++, C#, Rust, Java.

A COMPARATIVE ANALYSIS OF THE MOST POPULAR C-LIKE PROGRAMMING LANGUAGES

Research article

Bolshakov K.P.[1], Khaliman V.V.[2,] *
[2] ORCID : 0009-0007-4657-9440;
[1, 2] Far Eastern State Transport University, Khabarovsk, Russian Federation

* Corresponding author (vv.khaliman[at]mail.ru)

Abstract

The work presents a detailed comparative analysis of the most popular programming languages that have inherited C-like syntax: Rust, Go, C# and Java. Key characteristics, namely performance and resource efficiency, are examined. The supported programming paradigms and approach to the type system were analysed. The principles of memory management and the level of security provided (memory, multithreading) were also examined. Furthermore, ease of learning and the richness of the ecosystem were evaluated.

By comparing the pros and cons of each language for various tasks, a well-reasoned choice was made that offered the optimal balance of features, best suited to the specified technical requirements and the project's objectives.

Keywords: programming language, C-like languages, C, C++, C#, Rust, Java.

Введение

За длительное время существования электронных вычислительных машин было создано огромное количество разнообразных языков программирования. Каждый из них создан для решения своего набора задач, из-за чего каждый обладает как набором преимуществ, раскрывающихся в работе над поставленными задачами, так и рядом недостатков, обуславливающих малую популярность использования того или иного языка в других сферах.

Сейчас лидирующие позиции в рейтингах популярности занимают языки, подобные C, такие как C++, C#, Rust и Java [1]. В данной статье будут рассмотрены вышеперечисленные 5 языков программирования, включая C, а также проведён сравнительный анализ по ключевым характеристикам.

Вышеуказанные языки имеют схожую структуру кода, но имеют кардинальные отличия в наиболее важных характеристиках, таких как парадигма написания кода, типизация и методы управления памятью.

Языки C++, C#, Rust и Java относятся к СИ-подобным, так как имеют схожий синтаксис (блоки кода обозначаются фигурными скобками, после каждой инструкции ставится точка с запятой, операторы if, for, switch имеют схожую реализацию). Однако не все из них являются прямыми наследниками C, и у каждого языка есть и свои особенности, которые и определяют сферу, в которой он используется чаще всего.

Основная часть

Самым первым из этих языков был C, разработанный в 1972 году как замена ассемблеру. Он представляет компилируемый язык программирования общего назначения со статической типизацией. Он сочетает в себе высокую функциональность и производительность ассемблера, но гораздо проще в освоении и в адаптации для устройств с разной аппаратной базой. Его особенность заключается в том, что несмотря на функциональность, схожую с языками программирования высокого уровня, он работает очень близко к аппаратной части компьютера. Используя C, разработчик может напрямую взаимодействовать с памятью и регистрами процессора, что, с одной стороны, является

неоспоримым преимуществом, так как даёт возможность более эффективно распределять ресурсы устройства, но, с другой стороны, из-за отсутствия сборщиков мусора требуется каждый раз вручную очищать ресурсы [2].

В 1985 году появился прямой наследник C – C++. Это высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, поддерживающий разные парадигмы написания кода, как например функциональное, объектно-ориентированное и императивное. Обладает очень высокой производительностью, как у C, но имеет гораздо больший набор функций и библиотек, что позволяет использовать его под множество задач. Он, как и C, может напрямую взаимодействовать с ресурсами компьютера, но в отличие от предка в нём благодаря наличию ООП присутствует принцип RAII (Resource Acquisition Is Initialization). Его суть заключается в том, что если выделять память под объекты классов и контейнеры, а не просто под динамические массивы и переменные, то после выхода из зоны видимости программы объект неявно удаляется деструктором, и выделенная для него область памяти очищается [3].

C# и C — пример синтаксического наследования без прямой преемственности. C# не является прямым потомком C и кардинально от него отличается, но всё же унаследовал синтаксис.

C#, разработанный в 2002 году специально для платформы .NET, является компилируемым, объектно-ориентированным языком со статической типизацией. Благодаря развитию платформы .NET язык стал кроссплатформенным, хотя изначально был предназначен под написание приложений исключительно для операционной системы Windows. Он не так близко взаимодействует с аппаратной базой устройств и существенно медленнее C и C++, но всё ещё быстрее чем интерпретируемые языки и имеет положительную особенность – наличие JIT-компилятора [4].

JIT (Just-In-Time) — технология, позволяющая оптимизировать выполняемый код и обеспечить его переносимость между устройствами с разными процессорами. Его суть состоит в осуществлении динамической компиляции. Код компилируется не перед запуском программы, а непосредственно во время её выполнения, и не напрямую в машинный код, а сначала в байт-код, а затем уже в машинный, оптимизированный под конкретное устройство. Также JIT-компилятор может скомпилировать наиболее часто используемые участки кода для оптимизации временных затрат. Из-за этой технологии есть задержки при запуске больших приложений, но это компенсируется динамической оптимизацией кода и анализом частоты использования его блоков. Такой тип компиляции позволяет подготовить ПО к запуску на конкретном устройстве, обеспечивая кроссплатформенность [7].

Первая версия языка Java появилась 23 мая 1995 года в компании Sun Microsystems, впоследствии поглощённой компанией Oracle. Это объектно-ориентированный, статически типизированный, кроссплатформенный язык программирования общего назначения. Он популяризировал объектно-ориентированную парадигму программирования, предоставив простой, но мощный синтаксис, вдохновлённый C++, но исправив его недостатки в области управления памятью и сборщиков мусора. Это и сделало язык доступным для большого количества разработчиков, что привело к его популярности. Java уже долгие годы находится в тройке самых популярных языков программирования, на нём пишут корпоративные приложения, веб-сервисы (фреймворк Spring, апплеты), Android-приложения, работают с Big Data, IoT, ML [5].

Его ключевой особенностью является концепция «Write Once, Run Anywhere» («Напиши однажды, запускай везде»). Код Java можно запустить почти на любом устройстве. Такой уровень переносимости стал возможен благодаря JVM (Java Virtual Machine) — фундаментальной основой всей платформы Java. Виртуальная машина выступает посредником между байт-кодом и операционной системой, на которой исполняется этот код. Работа JVM состоит из нескольких критически-важных процессов:

1. Загрузка классов — находит и импортирует определения классов из файлов .class.
2. Верификация байт-кода — проверяет загруженный код на наличие ошибок и структурную целостность.
3. Выполнение кода — интерпретация программы JIT-компилятором, выявление «горячих» участков.
4. Управление памятью — автоматическая очистка памяти от неиспользуемых объектов с помощью сборщиков мусора.
5. Оптимизация — анализ и улучшение производительности кода во время выполнения.

JVM — это не конкретная программа, а название механизма. Существуют различные её реализации от разных разработчиков, каждая из которых может иметь свои особенности, но соответствует стандартной спецификации JVM. В свою очередь, виртуальная машина входит в JRE — Java Runtime Environment, куда также входят библиотеки классов, интеграционные компоненты для взаимодействия с операционной системой и инструменты базовой конфигурации и настройки среды выполнения. JRE — необходимый минимум для запуска Java-программ. Если же на устройстве требуется не только запускать программы, но и разрабатывать их, то необходимо установить JDK (Java Development Kit — набор инструментов для разработки). Он содержит в себе отладчик, архиватор, генератор документации, инструменты мониторинга и анализа производительности, а также дополнительные наборы библиотек [8].

Rust — компилируемый, строго типизированный язык программирования общего назначения, который так же, как и C++ делает упор на производительность и параллелизм. Изначально язык разрабатывался как личный проект Грейдона Хоара в 2006 году, когда он работал в Mozilla Research. В 2009 Mozilla официально начала спонсировать разработку и наняла инженеров для помощи Грейдону. До 2015 года Rust разрабатывался как внутренний проект, о котором мало кто знал и который нигде не использовался, пока он не стал выглядеть как потенциальный преемник C++. Именно тогда вышел первый релиз Rust 1.0 [9].

Сразу после официального релиза этот проект начал привлекать к себе внимание комьюнити, а позже и крупных компаний, заинтересованных в развитии проекта. Так в 2020 году был создан фонд Rust Foundation. В него вошли 5 компаний — Mozilla, AWS, Huawei, Google, Microsoft. Постепенно Microsoft стал переписывать различные

компоненты операционной системы Windows с C/C++ на Rust. Аналогично компания Google стала внедрять Rust в операционную систему Android. Постепенно Rust находит своё применение в Linux [6].

Одной из ключевых особенностей Rust является обеспечение строгих гарантий безопасности памяти. Rust предотвращает большое количество ошибок, таких как гонки данных и неправильное использование памяти. Достигается это благодаря двум мощным механизмам: системе владения (ownership) и системы заимствования (borrowing). Эти механизмы управления памятью предотвращают гонки данных, гарантируя, что они будут доступны только одному владельцу за раз и не позволяют одновременно считывать и изменять данные.

Кроме того, Rust проводит проверку времени жизни переменных и объектов, что гарантирует автоматическое освобождение памяти, когда время их жизни истечёт. Это предотвращает утечки памяти и другие проблемы, связанные с неправильным использованием памяти. Также язык предоставляет тип Option, который явно представляет значение, которое может отсутствовать, избавляя разработчиков от необходимости обрабатывать null-указатели и ошибки, связанные с ними [9].

В совокупности с современным синтаксисом и методами управления памятью, этот язык также обладает хорошей производительностью, сравнимой с показателями C и C++. Добиться этого удалось благодаря отсутствию сборщиков мусора и использованием LLVM. LLVM (Low Level Virtual Machine) — мощный инструмент оптимизации, интегрированный в компилятор Rust — rustc. Он представляет собой инфраструктуру для анализа, оптимизации и генерации кода, работающего на различных архитектурах процессоров и операционных системах. Также, в сходстве с C/C++, Rust способен работать близко к аппаратной части компьютера, что вместе с высокой производительностью дало возможность использовать этот язык в системном программировании. Также Rust нашёл своё применение в веб-разработке благодаря интеграции с JavaScript. На нём написаны игровые движки Amethyst и Bevy, которые дают достаточный инструментарий для создания игр различной сложности, от инди-проектов до AAA-игр [10]. В совокупности все вышеперечисленные факторы способствовали развитию сообщества и инфраструктуры вокруг языка, написанию большого количества библиотек и крупных проектов, а количество вакансий Rust-разработчиков с каждым годом будет только увеличиваться.

Все вышеупомянутые языки программирования являются универсальными и подходят для разных сфер деятельности, от создания игр до написания операционных систем и программного обеспечения для микроконтроллеров. Их главным преимуществом над другими языками является скорость выполнения программ, достигаемая благодаря близкому взаимодействию кода и устройства или специальным технологиям оптимизации. Но всё же у них есть некоторые различия. Итоговое сравнение приведено в таблице 1.

Таблица 1 - Сравнение языков программирования

DOI: https://doi.org/10.60797/COMP.2026.10.2.1

Язык	C	C++	C#	Java	Rust
Год создания	1972	1985	2002	1995	2015
Парадигма	Процедурный	Мульти-парадигменный	ООП	ООП	Мульти-парадигменный
Тип трансляции	Компиляция	Компиляция	JIT-компиляция (.NET Runtime)	JIT-компиляция (часть JVM)	Компиляция
Типизация	Статическая слабая	Статическая слабая (с шаблонами – сильная)	Статическая сильная	Статическая сильная	Статическая сильная
Управление памятью	Ручное	Ручное + RAII	Автоматическое (сборка мусора)	Автоматическое (сборка мусора)	Владение + заимствование
Безопасность потоков	Требует ручной синхронизации	Возможны гонки данных	Возможны гонки данных	Возможны гонки данных	Компилятор не даст скомпилировать небезопасный код
Скорость выполнения	Очень высокая	Очень высокая	Высокая	Высокая	Очень высокая (уровень C++)
Кроссплатформенность	Да (через перекомпиляцию)	Да (через перекомпиляцию)	Да (через JIT в .NET)	Да (JVM)	Да
Стандартная библиотека	Минимальная	Богатая (STL, алгоритмы, контейнеры)	Очень богатая (JDK)	Очень богатая	Небогатая (развивается)
Основное применение	ОС, драйверы	Игры, браузеры, высоконагруженное ПО	Приложения под Windows, Unity, Web	Enterprise, Android, Backend	Системное ПО, блокчейн, веб-сервисы

Благодаря своей многофункциональности, популярности и большому количеству образовательных материалов все вышеупомянутые языки хорошо подходят для изучения программирования, за исключением C. На данный момент этот язык устаревает, по нему становится всё меньше образовательных материалов, а большое количество устаревающего кода, который необходимо обслуживать, побуждает разработчиков к созданию новых технологий (например Rust), а не обслуживанию старых, в связи с чем язык программирования C в ближайшее время может потерять место в списке наиболее популярных.

Заключение

В результате сравнительного анализа пяти вышеупомянутых СИ-подобных языков программирования было выявлено, что каждый из них обладает своими преимуществами и ограничениями в зависимости от решаемых задач. В качестве наиболее подходящего для обучения языка был выбран Rust. Это обусловлено наличием у него эффективных и безопасных механизмов управления памятью и широким спектром возможных применений. Также, благодаря своей актуальности, он ещё долгое время будет развиваться и всё большее количество ПО будет переписано с более старых языков (C/C++) на новые, среди которых будет Rust.

Конфликт интересов

Не указан.

Рецензия

Все статьи проходят рецензирование. Но рецензент или автор статьи предпочли не публиковать рецензию к этой статье в открытом доступе. Рецензия может быть предоставлена компетентным органам по запросу.

Conflict of Interest

None declared.

Review

All articles are peer-reviewed. But the reviewer or the author of the article chose not to publish a review of this article in the public domain. The review can be provided to the competent authorities upon request.

Список литературы / References

1. PYPL PopularitY of Programming Language. — URL: https://pypl.github.io/PYPL.html (accessed: 05.02.2026).
2. Документация по C // Metanit. — URL: https://metanit.com/c/tutorial/1.1.php (дата обращения: 05.02.2026).
3. Документация по C++ // Metanit. — URL: https://metanit.com/cpp/tutorial/1.1.php (дата обраения: 05.02.2026).
4. Документация по C# // Metanit. — URL: https://metanit.com/sharp/tutorial/1.1.php (дата обращения: 05.02.2026).
5. Документация по Java // Metanit. — URL: https://metanit.com/java/tutorial/1.1.php (дата обращения: 17.01.2026).
6. Документация по Rust // Metanit. — URL: https://metanit.com/rust/tutorial/1.1.php (дата обращения: 05.02.2026).
7. Как работать с JIT // Habr. — URL: https://habr.com/ru/companies/badoo/articles/321378/ (дата обращения: 05.02.2026).
8. JVM, JDK, JRE, OpenJDK: разбираемся в основах Java-платформы // Sky.pro. — URL: https://sky.pro/wiki/media/raznicza-mezhdu-jvm-jdk-jre-i-openjdk/ (дата обращения: 05.02.2026).
9. Языки программирования, которые могут заменить C++ — Rust, Go, Swift, Carbon // Habr. — URL: https://habr.com/ru/companies/ first/articles/838752/ (дата обращения: 18.01.2026).
10. Язык программирования Rust: безопасность, производительность и преимущества // Habr. — URL: https://habr.com/ru/ articles/722658/ (дата обращения: 19.01.2026).

Список литературы на английском языке / References in English

1. PYPL PopularitY of Programming Language. — URL: https://pypl.github.io/PYPL.html (accessed: 05.02.2026).
2. Dokumentaciya po C [Documentation on C] // Metanit. — URL: https://metanit.com/c/tutorial/1.1.php (accessed: 05.02.2026). [in Russian]
3. Dokumentaciya po C++ [Documentation on C++] // Metanit. — URL: https://metanit.com/cpp/tutorial/1.1.php (accessed: 05.02.2026). [in Russian]
4. Dokumentaciya po C# [Documentation on C#] // Metanit. — URL: https://metanit.com/sharp/tutorial/1.1.php (accessed: 05.02.2026). [in Russian]
5. Dokumentaciya po Java [Documentation on Java] // Metanit. — URL: https://metanit.com/java/tutorial/1.1.php (accessed: 17.01.2026). [in Russian]
6. Dokumentaciya po Rust [Documentation on Rust] // Metanit. — URL: https://metanit.com/rust/tutorial/1.1.php (accessed: 05.02.2026). [in Russian]
7. Kak rabotat' s JIT [How to work with JIT] // Habr. — URL: https://habr.com/ru/companies/badoo/articles/321378/ (accessed: 05.02.2026). [in Russian]
8. JVM, JDK, JRE, OpenJDK: razbiraemsya v osnovah Java-platformy [JVM, JDK, JRE, OpenJDK:exploring the fundamentals of the Java platform] // Sky.pro. — URL: https://sky.pro/wiki/media/raznicza-mezhdu-jvm-jdk-jre-i-openjdk/ (accessed: 05.02.2026). [in Russian]
9. Yazyki programmirovaniya, kotorye mogut zamenit' C++ — Rust, Go, Swift, Carbon [Programming languages that can replace C++ — Rust, Go, Swift, Carbon] // Habr. — URL: https://habr.com/ru/companies/ first/articles/838752/ (accessed: 18.01.2026). [in Russian]
10. Yazyk programmirovaniya Rust: bezopasnost', proizvoditel'nost' i preimushchestva [The Rust Programming Language: security, performance, and benefits] // Habr. — URL: https://habr.com/ru/ articles/722658/ (accessed: 19.01.2026). [in Russian]