уведомлений пользователям системы, что значительно упрощает уведомление пользователей о предстоящих событиях в удобном формате.

Список использованных источников

1. Сайт «docs.spring.io». [Электронный ресурс]. – Режим доступа: https://docs.spring.io/spring-boot. – Дата доступа: 11.04.2025.
2. Сайт «core.telegram.org». [Электронный ресурс]. – Режим доступа: https://core.telegram.org/. – Дата доступа: 11.04.2025.

УДК 004.43

ЭВОЛЮЦИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Бернякович Н. И., студ., Кубасова А. А., студ., Соколова А. С., ст. преп.
Витебский государственный технологический университет,
г. Витебск, Республика Беларусь

Реферат. *Статья посвящена исследованию эволюции языков программирования. Рассмотрены ключевые этапы развития и влияние изменений в вычислительной технике на формирование современных технологий программирования.*

Ключевые слова: алгоритмы, Fortran, COBOL, C, Pascal, Smalltalk, C++, Java, Python, объектно-ориентированное программирование, функциональное программирование, доменно-специфичные языки.

Эволюция языков программирования является отражением человеческого стремления к автоматизации вычислительных процессов и постоянному поиску более эффективных способов решения задач в условиях быстро меняющихся технологических реалий. Истоки программирования можно проследить ещё до XIX века, когда Ада Лавлейс, вдохновлённая идеями Чарльза Бэббиджа, создала алгоритм для аналитической машины, который заложил фундаментальные принципы пошагового описания вычислений. Хотя в те времена вычислительная техника была далека от современных представлений, эта концепция стала предвестником того, что спустя более чем столетие привело к созданию специализированных языков, способных описывать сложные логические алгоритмы и процессы обработки данных.

С появлением электронных вычислительных машин в середине XX века понятие программирования приобрело совершенно новый смысл. Пионерские разработки, такие как Plankalkül Конрада Цузе, стали первыми попытками формализовать описание алгоритмов. В 1940-х годах Plankalkül предлагал элементы структурного программирования, задавая базовые принципы использования переменных, управляющих конструкций и даже рекурсии. Несмотря на ограниченность вычислительных возможностей того времени, эти идеи послужили катализатором для дальнейшего развития – появление ассемблеров позволило перейти от непосредственного обращения к машинным кодам к использованию мнемоник, что существенно сделало программирование более доступным и понятным.

В 1950-е годы с появлением первых языков программирования начинается новая эра в вычислительной технике. Язык Fortran, представленный в 1957 году, стал первым языком, который получил широкое применение в научных и инженерных расчетах. Он позволил программистам описывать сложные научные задачи посредством легко читаемого синтаксиса, что положительно сказалось на развитии вычислительной математики и физического моделирования. Параллельно с Fortran появились и другие специализированные языки – COBOL, ориентированный на обработку бизнес-данных, и LISP, разработанный для решения задач в области искусственного интеллекта. COBOL, созданный в 1959 году, предлагал упрощённую для понимания структуру и применялся в банковской, страховой и финансовой сферах, где требовалась надёжная обработка больших объёмов информации. Язык LISP, вышедший в 1958 году, был изначально нацелен на исследовательскую деятельность в сфере ИИ; его уникальная структура на основе S-выражений позволяла использовать один и тот

же формат для представления кода и данных, что открыло широкие возможности для метапрограммирования.

Переходя к периоду 1970–1980 годов, можно отметить, что этот этап ознаменовался качественными изменениями в парадигмах программирования. Новые требования к надежности, масштабируемости и структуры программного обеспечения привели к появлению языков, способствующих развитию объектно-ориентированного, структурного и процедурного программирования. Язык C, разработанный в 1972 году, открыл новую эру системного программирования, предоставив программистам возможность непосредственного управления памятью и ресурсами компьютера. Его использование в разработке операционных систем и компиляторов стало примером того, как низкоуровневый контроль может сочетаться с высокой производительностью. От языка C выросли последующие поколения языков, включая C++ и Java, что подчёркивает его стратегическую значимость.

Наряду с C активно развивались и языки, ориентированные на образовательные цели. Pascal, созданный в 1970 году, быстро завоевал популярность благодаря своей строгости, структурированному синтаксису и чёткой типизации. Он стал незаменимым инструментом для обучения алгоритмическому мышлению, позволяя студентам и начинающим программистам на ранних этапах осваивать базовые принципы построения программ. В этот же период появился и революционный Smalltalk, который ввёл понятие полностью объектно-ориентированного программирования. Благодаря концепциям классов, объектов, наследования и полиморфизма, Smalltalk продемонстрировал, как программное обеспечение может стать более модульным, гибким и легко масштабируемым. Его интерактивная среда разработки позволяла в реальном времени изменять и тестировать программный код, что стало важным опытом для последующего развития современных ООП-языков.

С началом 1990-х годов парадигмы разработки программного обеспечения претерпели значительные изменения, связанные с долгосрочным развитием объектно-ориентированных технологий. Язык C++, появившийся в 1983 году, объединил в себе возможности языка C и добавил поддержку объектно-ориентированного программирования, что позволило создавать более сложные и высокопроизводительные системы. Благодаря своей универсальности C++ быстро получил признание как в создании операционных систем, так и в разработке игровых движков и системных утилит. В свою очередь, язык Java, представленный в 1995 году, произвёл настоящую революцию в области кросс-платформенной разработки. Принцип «пиши один раз, запускай везде», реализованный посредством виртуальной машины Java (JVM), оптимизировал процесс разработки корпоративных и веб-приложений, обеспечив высокую степень безопасности и надёжности. Вместе с этими языками в 1991 году на свет появился Python, который благодаря интуитивно понятному синтаксису, динамической типизации и обширной библиотечной базе, стал универсальным инструментом для научных исследований, обработки данных и быстрой реализации прототипов.

В последние десятилетия современные тенденции в программировании характеризуются разнообразием и специализацией инструментов. На фоне стремительного развития облачных технологий, распределённых вычислительных систем и искусственного интеллекта, появились новые языки и фреймворки, ориентированные на нишевые задачи. Так, языки R и Julia получили широкое распространение в сфере анализа данных, статистического моделирования и машинного обучения благодаря богатству специализированных библиотек и удобным средствам визуализации. Для мобильной разработки популярны Swift и Kotlin, обеспечивающие высокую производительность и глубокую интеграцию с нативными платформами iOS и Android. В области системного программирования языки Go и Rust завоевывают доверие разработчиков за счёт высокой степени безопасности кода и эффективного управления ресурсами, что становится критически важным в создании отказоустойчивых систем.

Параллельно с этим активно развиваются направления функционального программирования, ранее преимущественно использовавшиеся в академической среде. Сегодня языки вроде Haskell, Scala, Erlang и Elixir находят применение в решении задач параллелизма, обработки больших объёмов данных и реализации чистых функций, позволяющих минимизировать ошибки в сложных вычислительных системах. Комбинирование функциональных методов с объектно-ориентированными и процедурными подходами даёт разработчикам возможность создавать ещё более надежные и гибкие решения, способные адаптироваться к постоянно меняющимся требованиям рынка. Особое внимание уделяется и вопросам безопасности программного обеспечения, где инструменты формальной верификации и автоматизации тестирования становятся неотъемлемой частью процесса разработки.

Будущее языков программирования, по всей видимости, будет определяться дальнейшей интеграцией различных парадигм и активным развитием доменно-специфичных языков (DSL). Такие языки позволят точнее решать специализированные задачи в областях биоинформатики, финтеха, интеллектуальных устройств и многих других, где важна не только скорость и производительность, но и высокий уровень надежности и безопасности. Помимо этого, ожидается, что дальнейшее развитие аппаратных средств и совершенствование алгоритмов будет способствовать созданию языков, которые смогут автоматически адаптироваться к новым условиям, оптимизировать ресурсы и минимизировать человеческий фактор при разработке программного обеспечения. В этом контексте важно отметить, что историческая динамика языков программирования всегда была тесно связана с изменениями в компьютерной технике и требованиями к методам решения вычислительных задач, и эта тенденция, скорее всего, сохранится в будущем.

Список использованных источников

1. Язык программирования [Электронный ресурс] // Википедия – свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Язык_программирования. – Дата доступа: 23.03.2025.
2. Эволюция языков программирования: основные этапы [Электронный ресурс] // Skypro Wiki. – Режим доступа: https://sky.pro/wiki/java/evolyuciya-yazykov-programmirovaniya-osnovnye-etapy/. – Дата доступа: 23.03.2025.
3. История языков программирования [Электронный ресурс] // Википедия – свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Историяязыковпрограммирования/. – Дата доступа: 23.03.2025.

UDC 004.415

APPLICATION OF MIXED REALITY TECHNOLOGY IN STUDENT LIFE SIMULATION GAMES

Li Ziyuan, master's degree student,
Dunina E., PhD in Physics and Mathematics, associate professor, Biziuk A., senior lecturer
Vitebsk State Technological University,Vitebsk, Republic of Belarus

Abstract. *Mixed reality or MR is a technology that combines both virtual and augmented reality When a person connects to MR, they find themselves in a mix of real and computer-generated worlds. Mixed reality technology is used in games, medicine, manufacturing, architecture, and education. The article discusses how mixed reality technology works.*

Keywords: mixed reality, information system, augmented reality, Meta Quest 3.

Mixed Reality (MR), as an important branch of extended reality (XR) technology, creates an interactive space where virtual and real are interwoven by dynamically integrating virtual objects with the real environment. Unlike augmented reality (AR), which only superimposes virtual information, MR emphasizes the two-way interaction between virtual content and the physical world. For example, virtual objects can perceive the light and shadow changes in the real environment, or provide real-time feedback on user actions. Compared with fully immersive virtual reality (VR), MR retains the user's ability to perceive the real environment, making the transition between the virtual and real boundaries more natural. This feature makes it unique in education and training, industrial design, medical simulation and other fields. In recent years, with the performance breakthroughs of head-mounted display devices – such as the high-precision spatial sensors, 4K display modules and low-latency tracking systems equipped in Meta Quest 3 - MR technology has gradually moved from the laboratory to large-scale applications. Its core value lies in the visualization of abstract concepts and the visualization of complex processes through a virtual-real symbiotic environment, thereby improving users' cognitive efficiency and practical ability.

In the field of education, traditional teaching often faces challenges of scene limitations and insufficient participation. For example, it is difficult for students to understand the working principles of three-