Содержание:

Введение

На сегодняшний день в жизни и деятельности людей очень важную роль играют информационные технологии. В настоящее время уже практически невозможно представить жизнь обыкновенного человека без компьютера и Интернета. Но для создания различных программ необходимы языки программирования.

Постоянный прогресс в развитии компьютерных технологий определяет необходимость появления разных новых знаковых систем для записи алгоритмов, то есть языков программирования. Смысл появления такого языка заключается в оснащенном наборе вычислительных формул дополнительной информации, который превращает в алгоритм данный набор. Языкам программирования свойственны перекликающиеся между собой цели: во-первых, они дают программисту аппарат для установки действий, которые должны быть выполнены, а во-вторых, формируются концепции, которые используются программистом, в размышлениях о том, что делать. Ответ на первую цель - язык, столь «близкий к машине», что все ключевые машинные аспекты могут быть легко и просто использованы для программиста очевидным образом. Ответ по второй цели - идеальным можно считать язык, который так «близок к решаемой задаче», чтобы концепция её решения могла быть выражена коротко и прямо. Это обуславливает актуальность выбранной темы.

В наше время C++ считается господствующим языком, его часто применяют для разработки коммерческих программных продуктов. Однако, в последние годы это господство начало колебаться, так как аналогичные претензии были заявлены таким языком как Java. В то же время общественное мнение резонировало, и многие программисты, бросившие C++ ради Java, все чаще возвращаются к прежней своей привязанности. При этом названные языки похожи [11].

Цель работы состоит в изучении истории возникновения и развития языков программирования Си (C++) и Java.

Для достижения установленной цели необходимо решить следующие задачи:

- изучить литературу по заданной теме;

- привести историю языков Си (C++) и Java;
- рассмотреть развитие языков Си (C++) и Java;

Объектом исследования выступают языки программирования Си (C++) и Java.

Предметом исследования - история, структура, польза, потенциал таких языков программирования, как Си (C++) и Java.

Структурно работа состоит из введения, двух глав, содержащих по два параграфа, заключения и списка использованной литературы.

Глава 1. История возникновения и развития языка программирования C++

1.1. Возникновение и эволюция языка C++

Бьерн Страуструп - разработчик языка C++ и создатель первого транслятора, сотрудник AT&T Bell Laboratories в Мюррей Хилл (Нью-Джерси, США) научно-исследовательского вычислительного центра. Он получил звание магистра математики и вычислительной техники в университете г. Аарус (Дания), а в кембриджском университете (Англия) докторское звание по вычислительной технике. Его специализация - распределенные системы, операционные системы, моделирование и программирование. Он стал автором полного руководства по языку C++ - «Руководство по C++ с примечаниями» вместе с М. А. Эллис [4].

Определенно C++ многим обязан языку C, который сохраняется в качестве его подмножества. Также сохранены и все присущие C средства низкого уровня, которые предназначены для решения самых насущных задач системного программирования. C, также, многим обязан предшествующему языку BCPL. Комментарий языка BCPL был восстановлен в C++. Еще один источник вдохновения - это язык SIMULA-67 – именно из него заимствована концепция классов (с производными классами и виртуальными функциями). Возможности перегрузки операций и свободы размещения описаний всюду, где может встречаться оператор в C++, напоминают язык Алгол-68 [5].

Предыдущие версии языка, которые назывались «C с классами», применялись с 1980 г. Данный язык возник, так как автору было необходимо написать программы

моделирования, управляемые прерываниями. Язык SIMULA-67 для этого идеально подходит, если не рассматривать эффективность. Язык «С с классами» применялся для больших задач моделирования. Тогда подверглись строгой проверке возможности написания на нем программ, для которых ресурсы памяти и времени критичны. В данном языке недоставало перегрузки операций, ссылок, виртуальных функций и многих иных возможностей. В июле 1983 г. С++ впервые покинул пределы исследовательской группы, где работал автор, однако многие возможности С++ тогда еще не были разработаны [9].

Само название С++ придумал Рик Маскитти летом 1983 г. Данное название как бы выделяет эволюционный прорыв в изменении языка С. Так как обозначение ++ описывает операцию инкремента С. Имя просто С+ представляет синтаксическую ошибку. Помимо этого, оно уже используется как название совершенно другого языка. Знатоки семантики С считают, что С++ слабее, чем ++С. Язык не назван D, так как он представляет расширение С, и в нем не предпринимается попыток решить какие-либо проблемы посредством отказа от возможностей С.

Сначала С++ был задуман для того, чтобы избежать программирования на ассемблере, С либо иных современных языках высокого уровня. Его главное предназначение заключается в упрощении и оптимизации процесса программирования для отдельного программиста. Еще не так давно не было на бумаге плана разработки С++. Проектирование, реализация и документирование шли параллельно. Не формировалось отдельного «проекта С++» либо «Комитета по разработке С++», поэтому язык развивается так, чтобы решить все проблемы, с которыми сталкиваются пользователи. Также толчками к развитию выступает обсуждение автором проблем языка с его коллегами и друзьями [8].

Со дня выхода в свет первого издания книги Страуструпа язык С++ неоднократно был подвержен серьезным уточнениям и изменениям. Преимущественно это затрагивает разрешение неоднозначности при перегрузке, связывание и управление памятью. Также, были внесены несущественные изменения для увеличения совместимости с языком С. Кроме того, были введены некоторые обобщения и существенные расширения, как, например: множественное наследование, функции-члены со спецификациями static и const, защищенные члены (protected), шаблоны типа и обработка особых ситуаций. Все названные расширения и доработки были нацелены на то, чтобы С++ стал языком, на котором можно создавать и использовать библиотеки.

Иные расширения, которые были введены в период 1985-1991 г.г. (такие как множественное наследование, статические функции-члены и чистые виртуальные функции), вероятнее появились как результат обобщения опыта программирования на C++, нежели были почерпнуты из иных языков.

Сделанные расширения языка за указанные шесть лет, прежде всего, были нацелены на рост выразительности C++ в качестве языка абстракции данных и объектно-ориентированного программирования вообще, и как средства для создания высококачественных библиотек в частности с пользовательскими типами данных.

Примерно в 1987 г. стало понятно, что работа по стандартизации C++ необходима и что нужно незамедлительно создавать основу для нее.

Основной вклад в эту работу внесла компания AT&T Bell Laboratories. Около сотни представителей из порядка двух десятков организаций изучали и комментировали то, что стало современной версией справочного руководства и исходными материалами для ANSI по стандартизации. C++. Как следствие, по инициативе компании Hewlett-Packard в декабре 1989 г. в составе ANSI был сформирован комитет X3J16. Прогнозируется, что работы по стандартизации C++ в ANSI (американский стандарт) станут составной частью работ по стандартизации силами ISO (Международной организации по стандартизации).

C++ развивался одновременно со становлением некоторых фундаментальных классов [10].

1.2. Структура и эффективность C++

C++ был развит на базе языка программирования C и сохраняет C как подмножество за очень немногими исключениями. Базовый язык, C подмножество C++, спроектирован таким образом, что сохраняется очень близкое соответствие между его типами, операторами и операциями, и компьютерными объектами, с которыми приходится иметь дело: символами, числами и адресами. Исключением выступают операции свободной памяти new и delete, отдельные операторы и выражения C++ как правило не нуждаются в скрытой поддержке во время выполнения или подпрограммах.

В C++ применяются такие же последовательности возврата и вызова из функций, как и в C. В случаях, когда даже приведенный достаточно эффективный механизм становится слишком дорогим, C++ функция может быть подставлена inline, таким образом, удовлетворяя соглашению о записи функций без дополнительных расходов времени выполнения.

Одно из начальных предназначений C – это использование его вместо программирования на ассемблере в насущных задачах системного программирования. При проектировании C++, принимались меры, чтобы успехи в этой области не попадали под угрозу. Отличие между C и C++ в первую очередь состоит в степени внимания, которое уделяется типам и структурам. C снисходителен и выразителен. C++ еще более выразителен, но, чтобы добиться этой выразительности, программисту необходимо уделить особое внимание типам объектов. Когда типы объектов известны, компилятор может верно обрабатывать выражения, однако в противном случае программисту нужно было бы задавать действия с мельчайшими подробностями. Знание типов объектов также дает возможность компилятору выявить ошибки, которые могли остаться до тестирования в противном случае. Нужно отметить, что применение системы типов для получения проверки параметров функций, защиты данных от случайного искажения, задания новых операций и т.д., само по себе не вызывает рост расходов по времени выполнения и памяти.

Особенное внимание, уделенное структуре в ходе разработки C++, отражается на росте масштаба программ, которые были написаны со времени разработки C. Небольшую программу (меньше 1000 строк) можно заставить работать при помощи грубой силы, даже обходя все правила хорошего стиля. Но это не совсем так для программ больших размеров. Если программа в 10 000 строк обладает плохой структурой, то можно заметить, что новые ошибки возникают так же быстро, как старые удаляются. C++ был разработан так, чтобы позволить структурировать большие программы разумным образом так, чтобы для одного человека не было невозможным справляться с программами в 25 000 строк. Существуют и программы, куда больших размеров, однако работающие, в общем, как правило, состоят из большого количества практически независимых частей, размер каждой из них намного ниже определенных пределов. Как правило, сложность написания и поддержки программы также зависит и от сложности разработки, а не только от количества строк текста программы.

Однако, не каждая часть программы может быть верно структурирована, независима от аппаратного обеспечения, легко читаема и пр. C++ имеет

возможности, предназначенные для непосредственной и эффективной работы с аппаратными средствами, не беспокоясь о безопасности либо простоте понимания. Также он обладает возможностями, позволяющими скрывать такие программы за надежными, но элегантными интерфейсами.

Может сложится мнение, что спецификация программы при помощи более подробной системы типов приведет к увеличению размеров исходных текстов программы. В C++ это не так. C++ программа, которая описывает типы параметров функций, использует классы и т.д., как правило, немного короче эквивалентной C программы, в которой данные средства не применяются [3].

Глава 2. История возникновения и развития языка программирования Java

2.1. Возникновение и эволюция языка Java

Язык Java зародился в качестве части проекта создания передового программного обеспечения для разных бытовых приборов. Проект начали реализовать на языке C++, но вскоре обнаружился ряд проблем, лучшее средство борьбы с которыми - это изменение непосредственно инструмента, а именно языка программирования. Было очевидно, что нужен платформенно-независимый язык программирования, который позволял бы создавать программы, не нуждающиеся в компиляции отдельно для каждой архитектуры и так же возможно было бы использование на разных процессорах под разными операционными системами [12].

Появлению языка Java предшествовала достаточно интересная история. В 1990 году разработчик ПО компании Sun Microsystems Патрик Нотон понял, что его тяготит необходимость осуществления поддержки сотни разных интерфейсов программ, применяемых в фирме, и уведомил исполнительного директора Sun Microsystems и своего друга Скотта Макнили о своем решении перейти работать в фирму NeXT. Макнили, попросил Нотона сформировать список причин такого недовольства и попробовать выдвинуть такое решение проблем, как если бы он мог исполнить все, что угодно.

Нотон, хоть и не особо рассчитывал, что кто-то в серьез обратит внимание на его письмо, все же выдвинул свои претензии, прямо раскритиковав недостатки Sun

Microsystems, в том числе, разрабатываемую как раз в тот момент архитектуру ПО NeWS. К удивлению Нотона, его письмо имело оглушительный успех: его разослали всем ведущим инженерам Sun Microsystems, которые, в свою очередь, незамедлительно откликнулись и выказали одобрение и поддержку своему коллеге. Обращение также вызвало одобрение и у высшего руководства фирмы, а именно, у Билла Джоя, основателя Sun Microsystems, и Джеймса Гослинга (James Gosling), начальника Нотона.

В день, когда Нотон должен был покинуть компанию, было принято решение о формировании команды ведущих разработчиков с условием, чтобы они делали что угодно, лишь бы создали нечто необыкновенное. Команда в составе шести человек начала разработку нового объектно-ориентированного языка программирования, который был назван Oak (дуб), в честь дерева, которое росло под окном Гослинга [1].

Затем компания Sun Microsystems команду Green преобразовала в компанию First Person. Молодая компания имела интереснейшую концепцию, но подходящего ей применения не могла найти. После нескольких неудач ситуация для фирмы неожиданно резко изменилась: был анонсирован браузер Mosaic. Таким образом родился World Wide Web, который положил начало бурному развитию Internet. Нотоном было предложено использовать Oak для создания Internet-приложений. Так Oak стал самостоятельным продуктом, затем был написан Oak-компилятор и Oak-браузер "WebRunner". В 1995 году компания Sun Microsystems решила все же объявить о новом продукте, только переименовала его в Java. Когда же Java оказалась в руках Internet, стало ясно, что необходимо запускать Java-аплеты, они представляют собой небольшие программы, которые загружаются через Internet. WebRunner в следствие переименовали в HotJava и фирма Netscape стала поддерживать Java-продукты [6].

Java создавался как универсальный язык, который предназначался для прикладного программирования в неоднородных компьютерных сетях и со стороны клиентского компьютера, и со стороны сервера. В том числе – для применения на тонких аппаратных клиентах (устройствах малой вычислительной мощности с крайне ограниченными ресурсами). При этом скомпилированные программы Java работают лишь под управлением виртуальной Java-машины, именно поэтому они называются приложениями Java. Синтаксис операторов Java почти полностью идентичен синтаксису языка C, но, Java не является расширением C, в отличие от C++, – это совершенно независимый язык, со своими собственными синтаксическими правилами.

Идеология Java подразумевает работу в компьютерных сетях и возможность подгрузки в нужный момент посредством сети требуемых классов и ресурсов, необходимых программе, и тех, что не были загружены до того. Для обеспечения такой работы приложения Java разрабатываются и распространяются, как большое число независимых классов. Тем не менее, данный способ разработки ведет к очень высокой фрагментации программы. Даже небольшие учебные проекты, как правило, состоят из десятков классов, в то время как реальные проекты – из сотен. В то же время каждому общедоступному (public) классу соответствует свой файл, носящий такое же имя. Для того чтобы справиться с данным количеством файлов, Java предусматривает специальное средство группировки классов, которое называется пакетом (package). Пакеты обеспечивают независимые пространства имен (namespaces), а также ограничение доступа к классам [2].

2.2. Этапы развития языка Java и его применение

Этапы разработки языка Java, изложенные Патриком Нотоном, соавтором браузера HotJava и нынешним вице-президентом по технологии корпорации Starwave.

5 декабря 1990 г. - Нотон отклоняет предложение перейти в фирму NeXT и начинает работу в фирме Sun над проектом, который впоследствии получил название Green.

15 января 1991 г. – Совещание по типу «мозговой штурм» по проекту Stealth в Аспене, где принимали участие Билл Джой, Уэйн Розинг, Энди Бехтолсхейм, Майк Шеридан, Патрик Нотон и Джейм Гослинг.

1 февраля 1991 г. – Шеридан, Гослинг и Нотон основательно берутся за работу. Нотон отвечает за графическую систему Aspen, Гослинг - идеи языка программирования, Шеридан - бизнес-разработка.

8 апреля 1991 г. - Переезд на новый адрес и обрыв прямого соединения с локальной сетью (как и большинством других средств связи) фирмы Sun; проект разрабатывается под названием Green.

15 апреля 1991 г. - К проекту Green присоединяются Крейг Форрест (дизайнер чипа SS10), Эд Фрэнк (архитектор системы SPARCstation 10) и Крис Уорт (разработчик системы NeWS).

Май 1991 г. - Эд Фрэнк нарекает прототип аппаратуры названием *7 (или Star7; *7 - код, который было необходимо набрать в офисе Sand Hill, для того, чтобы ответить на любой звонок с любого телефона).

Июнь 1991 г. - Гослинг занимается разработкой интерпретатора Oak, который в дальнейшем (при поисках торговой марки) переименовали в Java.

1 августа 1991 г. - Произошло объединение Oak и Aspen; запуск их первой реальной программы.

19 августа 1991 г. - Разработчики Green демонстрирует концепцию базового пользовательского интерфейса и графической системы сооснователям компании Sun Биллу Джою и Скотту Макнили.

17 октября 1991 г. - Нотон и Шеридан присваивают девиз "1st Person" конструкторской философии своего коллектива, со временем данный девиз становится названием компании.

17 ноября 1991 г. - Офис проекта Green вновь подключается к основной сети компании Sun линией на 56 Кбит/с.

1 марта 1992 г. - К проекту Green присоединяется Джонатан Пейн, позднее участвовавший в написании HotJava.

Лето 1992 г. - Активная деятельность, направленная на доработку Oak, пользовательского интерфейса, Green OS, аппаратуры Star7 и соответствующих компонентов.

4 сентября 1992 г. - Завершение разработки устройства Star7; демонстрация устройства Макнили и Джою.

1 октября 1992 г. - Уэйн Розинг переходит из фирмы SunLabs, и принимает руководство коллективом на себя.

1 ноября 1992 г. - Организована корпорация FirstPerson.

15 января 1993 г. - Коллектив переезжает в Пало Альто в здание, где ранее находилась лаборатория Western Research Lab компании DEC и была основана начальная группа Hamilton Group (она же OSF).

15 марта 1993 г. - После анализа результатов испытаний кабельного интерактивного телевидения, которые были проведены компанией Time Warner,

корпорация FirstPerson переключается на эту тематику.

Апрель 1993 г. - Выпуск первого графического браузера для Internet - Mosaic 1.0, разработанного в центре NCSA.

14 июня 1993 г. - Компания Time Warner продолжает проводить испытания интерактивного кабельного ТВ с компанией SGI, не обращая внимание на очевидное превосходство технологии компании Sun и уверения, что Sun выиграла эту сделку.

Лето 1993 г. - Нотон преодолевает 300 тыс. миль, пытаясь продать Oak всем, кто занимается интерактивным телевидением и бытовой электроникой; в это время темп, с которым люди получают доступ к Internet, невероятно нарастает.

Август 1993 г. - Спустя несколько месяцев серьезных переговоров с компанией 3DO касательно разработки ОС для приставок, президент 3DO Трип Хокинс делает предложение купить технологию. Макнили отказывается, и сделка срывается.

Сентябрь 1993 г. - К коллективу присоединяется Артур Ван Хофф, поначалу - чтобы создать среду разработки приложений, которые предназначены для интерактивного телевидения, а потом и сам язык.

7 декабря 1993 г. - Экспертиза операций на высоком уровне в FirstPerson выясняет, что данная группа не имеет реальных партнеров либо маркетинговой стратегии и совершенно неясно представляет себе дату выпуска.

8 февраля 1994 г. - Отменено публичное заявление фирмы FirstPerson о выпуске, которое было намечено на конференцию Technology, Entertainment and Design (TED).

17 февраля 1994 г. - Для разносторонней экспертизы исполнительным лицам компании Sun представлен альтернативный бизнес-план по разработке мультимедийной платформы для CD-ROM и онлайновой работы корпорации FirstPerson.

25 апреля 1994 г. - Создана компания Sun Interactive; туда перемещается половина сотрудников FirstPerson.

Июнь 1994 г. - Начат проект Liveoak, который нацелен Биллом Джоем на применение Oak в крупном проекте небольшой операционной системы.

Июль 1994 г. - Нотон ограничивает сферу применения проекта Liveoak, просто поменяв ориентацию Oak на Internet.

16 сентября 1994 г. - Нотон и Пейн начинают писать WebRunner – браузер по типу Mosaic, позже переименованный в HotJava.

29 сентября 1994 г. - Прототип HotJava впервые продемонстрирован исполнительным лицам компании Sun.

11 октября 1994 г. - Нотон переходит в компанию Starwave.

Осень 1994 г. - Ван Хофф реализует компилятор Java на языке Java. (Ранее Гослинг реализовывал его на языке C).

23 мая 1995 г. - Компания Sun официально представляет Java и HotJava на выставке SunWorld '95.

Компания Sun Microsystems выпускает первую версию Java в начале 1996 г. Однако, пользователи быстро осознали, что версия Java 1.0 не подходит для разработки серьезных приложений. Конечно, данную версию можно применить для реализации визуальных эффектов на веб-страницах, к примеру, написать апплет, который выводит на страницу случайно "прыгающий" текст, но версия Java 1.0 была еще сырой. В ней даже не было предусмотрено средства вывода на печать. Можно сказать, что версия Java 1.0 еще не была готова. В последующей версии, Java 1.1, были устранены наиболее серьезные недостатки, улучшены средства рефлексии и реализована новая модель событий для программирования ГПИ. Тем не менее, ее возможности все еще были ограничены.

Выпуск версии Java 1.2 стал главной новостью на конференции JavaOne в 1998 г. В новой версии слабые средства реализации графического пользовательского интерфейса и графических приложений были заменены мощным инструментарием. Это был большой шаг вперед, к соответствию лозунгу «Write Once, Run Anywhere» («Однажды написано — везде выполняется»), который был выдвинут при разработке предыдущих версий. В декабре 1998 года, через три дня после выхода в свет, название новой версии было изменено на неудобное и длинное Java 2 Standard Edition Software Development Kit Version 1.2 (стандартная редакция набора инструментальных средств для разработки программного обеспечения на Java 2, версия 1.2).

Помимо Standard Edition, также были предложены следующие два варианта: для портативных устройств Micro Edition («микроредакция»), к примеру, для мобильных телефонов, и Enterprise Edition (редакция для корпоративных приложений).

Версии 1.3 и 1.4 набора инструментальных средств Standard Edition - результаты поэтапного усовершенствования первоначально выпущенной версии Java 2. Они имеют новые возможности, повышенную производительность и, естветственно, содержат намного меньше ошибок. В ходе развития Java многие взгляды на апплеты и клиентские приложения были пересмотрены. В частности, оказалось, что на Java удобно разрабатывать высококачественные серверные приложения.

В версии 5.0 язык Java подвергся наиболее серьезной модификации с момента выпуска версии 1.1. (первоначально версия 5.0 имела номер 1.5, но на конференции JavaOne в 2004 г. была принята новая нумерация версий.) После многолетних исследований были добавлены обобщенные типы (которые приблизительно соответствуют шаблонам C++), хотя при этом не были выдвинуты требования модификации виртуальной машины. Ряд других языковых элементов, например, циклы в стиле for each, автоупаковка и метаданные, были явно навеяны языком C#.

Версия 6 (без суффикса .0) была выпущена в конце 2006 г. И снова сам язык существенных изменений не претерпел, но были внесены усовершенствования, которые были связаны с производительностью, а также произведены расширения библиотек.

По мере того как в центрах обработки данных все чаще стали применяться аппаратные средства широкого потребления вместо специализированных серверов, для компании Sun Microsystems наступили тяжелые времена, и в конечном итоге она была приобретена компанией Oracle в 2009 г. Разработка следующих версий Java на долгое время приостановилась. И лишь в 2011 году компания Oracle выпустила новую версию Java 7 с простыми усовершенствованиями. Было решено более серьезные изменения отложить до версии Java 8, выпуск которой произошел в 2014 г.

В табл. 1 сведены данные об этапах развития языка и библиотек Java [7].

Таблица 1

Этапы развития языка Java

Версия	Год выпуска	Новые языковые средства	Количество классов и интерфейсов
1.0	1996	Выпуск самого языка	211
1.1	1997	Внутренние классы	477
1.2	1998	Отсутствуют	1524
1.3	2000	Отсутствуют	1840
1.4	2002	Утверждения	2723
5.0	2004	Обобщенные классы, цикл в стиле for each, автоупаковка, аргументы переменной длины, метаданные, перечисления, статический импорт	3279
6	2006	Отсутствуют	3793
7	2011	Оператор switch со строковыми метками ветвей, ромбовидный оператор, двоичные литералы, усовершенствованная обработка исключений	4024
8	2014	Лямбда-выражения, библиотеки потоков и даты/времени, интерфейсы с методами по умолчанию	4240

9	2017	Литералы в коллекциях, оператор Элвиса, Class Optional, Streams, IO, Regrexp, обработка процессов ProcessHandle	более 4500

В 2018 году коренным образом изменилась парадигма разработки и выпуска релизов Java.

Только в конце марта текущего года, вышел Java 10. Однако, в связи с тем, что компания Oracle изменила релизный цикл (каждые полгода новый релиз), к выходу уже готовится 11-я версия.

Java 10: краткая сводка.

Нововведения десятой версии: локальный вывод типов с помощью var, улучшения в процессах «сборки мусора» и возможность использовать Graal как основной JIT-компилятор.

Локальный вывод типов с помощью var просили многие разработчики. Теперь можно не вводить типы два раза подряд: сперва для объявления переменной, а затем для конструктора, идущего следом.

var list = new ArrayList<String>(); // infers ArrayList<String>

var stream = list.stream(); // infers Stream<String>

Тем не менее, решение ввести var закончилось неоднозначными отзывами участников сообщества. Кто-то поддерживал нововведение и утверждал, что дедупликация повышает читабельность кода. Но были и те, кто уверял, что сейчас ряд типов переменных (к примеру, connection) не будут очевидными. Несмотря на что, что IDE теперь могут показывать их по требованию, однако же, в других средах будут возникать проблемы.

В десятом релизе для улучшенной сборки мусора содержались сразу два изменения: JEP 304 и JEP 307. JEP 304 улучшил изоляцию кода от разных сборщиков мусора посредством нового интерфейса GC, что дало возможность быстрее интегрировать сторонние сборщики. JEP 307, в свою очередь, позволил «собирать мусор» в несколько потоков.

Что можно сказать о новом JIT-компиляторе, то он направлен на рост производительности JVM. Предыдущая версия JVM была написана на C++, но в рамках проекта Metropolis значительную часть JVM перепишут на Java. Экспериментальный компилятор выступает первым шагом на пути к данной цели.

Возможные нововведения в Java 11. В начале осени разработчики планируют представить Java 11. И о функциях, которые могут быть частью релиза, уже известно. В сети происходит бурное обсуждение вероятных изменений.

Часть разработчиков недовольна тем, как быстро меняется язык. Но есть и те, кто отмечает, что Java наконец-то обзаводится функциями, которых ему не хватало, и которые уже давно были реализованы в других ЯП. Вот некоторые из них.

Изменения в локальном выводе типов.

Java 10 уже предоставил возможность использовать var для обозначения типа локальных переменных, перекладывая эту задачу на плечи компилятора. Однако Java 11 идет дальше и делает так, что var можно использовать при объявлении формальных параметров неявно типизированных лямбда-выражений.

Добавление необработанных строковых литералов.

Это еще одно дополнение, над которым сейчас ведется работа. В необработанной строке каждый символ читается «как есть», включая символы разрыва. Например, такой строкой может быть разметка HTML или SQL-запрос:

String html = "<html>\n" +

" <body>\n" +

" Hello World. \n" +

" </body>\n" +

"</html>\n";

Используя строковый литерал, этот код можно записать следующим образом:

String html = `<html>

<body>

Hello World.

</body>

</html>

`;

Для обозначения необработанной строки используется обратный апостроф (`). Если вам нужно прописать апостроф внутри самой строки, то в этом случае для маркировки её границ используются двойные (или тройные, четверные и т. д.) обратные апострофы:

String str = ```This is a raw `string` with ``backticks`` inside```;

Появятся switch-выражения

Сейчас оформление switch выполняется следующим образом:

int val;

switch (str) {

case "foo": val = 1; break;

case "bar": val = 2; break;

case "baz": val = 3; break;

default: val = -1;

}

С появлением switch-выражений, конструкция сократится:

int val = switch (str) {

case "foo": break 1;

case "bar": break 2;

case "baz": break 3;

default: break -1;

}

При этом если в case встречается только break, возможно использовать упрощенную нотацию:

int val = switch (str) {

case "foo" -> 1;

case "bar" -> 2;

case "baz" -> 3;

default -> -1;

}

Добавление switch expressions — это шаг вперед на пути к появлению метода сопоставления с образцом.

Помимо отмеченных выше, Java может получить и другие изменения. Например, поддержку типов значений или переменных типа в enum. Эти обновления не обязательно войдут в состав Java 11, но, вероятно, ожидать их можно уже в ближайшем будущем [13].

Заключение

C++ был развит на базе языка программирования C и сохраняет C как подмножество за очень немногими исключениями. Базовый язык, C подмножество C++, спроектирован таким образом, что сохраняется очень близкое соответствие между его типами, операторами и операциями, и компьютерными объектами, с которыми приходится иметь дело: символами, числами и адресами. Исключением выступают операции свободной памяти new и delete, отдельные операторы и выражения C++ как правило не нуждаются в скрытой поддержке во время выполнения или подпрограммах.

В C++ применяются такие же последовательности возврата и вызова из функций, как и в C. В случаях, когда даже приведенный достаточно эффективный механизм становится слишком дорогим, C++ функция может быть подставлена inline, таким образом, удовлетворяя соглашению о записи функций без дополнительных расходов времени выполнения.

Java создавался как универсальный язык, который предназначался для прикладного программирования в неоднородных компьютерных сетях и со стороны клиентского компьютера, и со стороны сервера. В том числе – для применения на тонких аппаратных клиентах (устройствах малой вычислительной мощности с крайне ограниченными ресурсами). При этом скомпилированные программы Java работают лишь под управлением виртуальной Java-машины, именно поэтому они называются приложениями Java. Синтаксис операторов Java почти полностью идентичен синтаксису языка C, но, Java не является расширением C, в отличие от C++, – это совершенно независимый язык, со своими собственными синтаксическими правилами.

Идеология Java подразумевает работу в компьютерных сетях и возможность подгрузки в нужный момент посредством сети требуемых классов и ресурсов, необходимых программе, и тех, что не были загружены до того. Для обеспечения такой работы приложения Java разрабатываются и распространяются, как большое число независимых классов. Тем не менее, данный способ разработки ведет к очень высокой фрагментации программы. Даже небольшие учебные проекты, как правило, состоят из десятков классов, в то время как реальные проекты – из сотен. В то же время каждому общедоступному (public) классу соответствует свой файл, носящий такое же имя. Для того чтобы справиться с данным количеством файлов, Java предусматривает специальное средство группировки классов, которое называется пакетом (package). Пакеты обеспечивают независимые пространства имен (namespaces), а также ограничение доступа к классам.

Список использованной литературы

1. Бондарев В.М. Учебное пособие по программированию на Java. - Харьков: СМИТ, 2013. - 296 с.
2. Гаврилов А.В. Учебное пособие по языку Java. - Самара: Изд-во СГАУ, 2010. - 175 с.
3. Голицына О.Л. Языки программирования. Учебное пособие. - М.: Форум, Инфра-М, 2008. - 125 с.
4. Дейтел П.Дж., Дейтел Х.М. Как программировать на C++. Введение в объектно-ориентированное проектирование с использованием UML. / Пер. с англ. - М.: Издательство Бином, 2009. - 1454 с.
5. Дрейер М. C# для школьников. - М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2009. - 128 с.

6. Дубаков А.А. Введение в объектно-ориентированное программирование на Java. – СПб: Университет ИТМО, 2016. – 248 с.
7. Кузнецов В.В. Современное программирование на Java. - Томск, 2014. - 292 с.
8. Макаров В.Л. Программирование и основы алгоритмизации. - СПб.: СЗТУ, 2003. - 110 с.
9. Пышкин Е.В. Структурное проектирование: основание и развитие методов. С примерами на языке C++. – СПб.: Изд-во Политехнического ун-та, 2005. – 324 с.
10. Радченко Г.И. Объектно-ориентированное программирование / Г.И. Радченко, Е.А. Захаров. - Челябинск: Издательский центр ЮУрГУ, 2013. - 167 с.
11. Тюгашев А.А. Основы программирования. – СПб: Университет ИТМО, 2016. – 160 с.
12. Хабибуллин И.Ш. Самоучитель Java 2. СПб.: БХВ-Петербург, 2007. - 720 с.
13. Шилдт Герберт, Холмс Джеймс. Искусство программирования на Java. - СПб.: Питер, 2005. - 336 с.