Содержание:

ВВЕДЕНИЕ

В настоящее время современный человек не может представить свою жизнь без телевизора, радио, сотового телефона, а главное без компьютера. В современном мире персональный компьютер играет главную роль во всех сферах деятельности, во всех странах мира. Конечно, можно отметить, что еще несколько десятилетий назад, никто не пользовался ПК и при этом не имели никаких удобств, но мир не стоит на месте и необходимо идти «в ногу со временем».

Высокоуровневый язык программирования — язык программирования, разработанный для быстроты и удобства использования программистом. Основная черта высокоуровневых языков — это абстракция, то есть введение смысловых конструкций, кратко описывающих такие структуры данных и операции над ними, описания которых на машинном коде (или другом низкоуровневом языке программирования) очень длинны и сложны для понимания. В этом и заключается актуальность данной работы. Очень часто такие языки используются в последнее время в веб-разработка, потому что разработка веб-приложений находится на рекордно высоком уровне, поскольку все больше и больше людей получают доступ к сети, а также рост многих веб-компаний приводит к популяризации этих технологий.

Веб-приложения — это приложения, которые запускаются в веб-браузере, и именно поэтому они популярны из-за повсеместности веб-браузеров. Это также удобно из-за того, что не нужно устанавливать программное обеспечение или приложение на десятки, сотни или даже тысячи компьютеров, что, в свою очередь, упрощает обслуживание приложения. В этом и заключается актуальность данной работы.

Целью данной работы является обзор языков программирования высокого уровня.

Объектом исследования являются языки программирования, а предметом исследования обзор высокоуровневых языков программирования.

В рамках работы необходимо решить следующие задачи:

- изучить соответствующую литературу;

- изучить историю появления «программирования»;
- проанализировать низкоуровневые и высокоуровневые языки программирования;
- провести анализ высокоуровневых языков программирования;
- сделать выводы по проделанной работе.

Для решения поставленной задачи была изучена литература в области программирования высокоуровневых языков. Список литературы представлен в последнем разделе.

1. Анализ предметной области

1.1 История появления программирования

Одной из самых революционных идей, приведших к созданию автоматических цифровых вычислительных машин, была высказанная в 20-х годах 19 века Чарльзом Бебиджем мысль о предварительной записи порядка действия машины для последующей автоматической реализации вычислений – программе. И, хотя использованная Бебиджем запись программы на перфокартах, придуманная для управления такими станками французским изобретателем Жозефом Мари Жаккаром, технически не имеет ничего общего с современными приемами хранения программ в ПК, принцип здесь по существу один. С этого момента начинается история программирования.

Аду Левлейс, современницу Бебиджа, называют первым в мире программистом. Она теоретически разработала некоторые приемы управления последовательностью вычислений, которые используются в программировании и сейчас. Ею же была описана и одна из важнейших конструкций практически любого современного языка программирования – цикл.

Революционным моментом в истории языков программирования стало появление системы кодирования машинных команд с помощью специальных символов, предложенной Джоном Моучли.

Система кодирования, предложенная им, вдохновила одну из его сотрудниц Грейс Мюррей Хоппер. При работе на компьютере «Марк-1» ей и ее группе пришлось столкнуться со многими проблемами и все, что ими придумано, было впервые. В

частности, они придумали подпрограммы. И еще одно фундаментальное понятие техники программирования впервые ввели Хоппер и ее группа – «отладка». [12, 19]

В конце 40-х годов Дж. Моучли создал систему под названием «Short Code», которая являлась примитивным языком программирования высокого уровня. В ней программист записывал решаемую задачу в виде математических формул, а затем, используя специальную таблицу, переводил символ за символом, преобразовывал эти формулы в двухлитерные коды. В дальнейшем специальная программа компьютера превращала эти коды в двоичный машинный код. Система, разработанная Дж. Моучли, считается одним из первых примитивных интерпретаторов.

Уже в 1951 г. Хоппер создала первый в мире компилятор и ею же был введен сам этот термин. Компилятор Хоппер осуществлял функцию объединения команд и в ходе трансляции производил организацию подпрограмм, выделение памяти компьютера, преобразование команд высокого уровня (в то время псевдокодов) в машинные команды. «Подпрограммы находятся в библиотеке (компьютера), а когда вы подбираете материал из библиотеки – это называется компиляцией» – так она объясняла происхождение введенного ею термина.

В 1954 году группа под руководством Г. Хоппер разработала систему, включающую язык программирования и компилятор, которая в дальнейшем получила название Math-Matic. После удачного завершения работ по созданию Math-Matic Хоппер и ее группа принялись за разработку нового языка и компилятора, который позволил бы пользователями программировать на языке, близком к обычному английскому. В 1958 г. появился компилятор Flow-Matic. Компилятор Flow-Matic был первым языком для задач обработки коммерческих данных. [4, 18]

Разработки в этом направлении привели к созданию языка Кобол (COBOL – Common Business Oriented Language). Он был создан в 1960 году. В этом языке по сравнению с Фортраном и Алголом, слабее развиты математические средства, но зато хорошо развиты средства обработки текстов, организация вывода данных в форме требуемого документа. Он задумывался как основной язык для массовой обработки данных в сферах управления и бизнеса.

Середина 50-х годов характеризуется стремительным прогрессом в области программирования. Роль программирования в машинных командах стала уменьшаться. Стали появляться языки программирования нового типа, выступающие в роли посредника между машинами и программистами. Первым и

одним из наиболее распространенных был Фортран (FORTRAN, от FORmula TRANslator – переводчик формул), разработанный группой программистов фирмы IBM в 1954 году (первая версия). Этот язык был ориентирован на научно-технические расчеты математического характера и является классическим языком программирования при решении на ПК математических и инженерных задач.

Для первых языков программирования высокого уровня предметная ориентация языков была характерной чертой.

Особое место среди языков программирования занимает Алгол, первая версия которого появилась в 1958 году. Одним из разработчиков Алгола был «отец» Фортрана Джон Бэкус. Название языка ALGOrithmic Language подчеркивает то обстоятельство, что он предназначен для записи алгоритмов. Благодаря четкой логической структуре Алгол стал стандартным средством записи алгоритмов в научной и технической литературе. [4, 16]

В середине 60-х годов Томас Курц и Джон Камени (сотрудники математического факультета Дартмунтского колледжа) создали специализированный язык программирования, который состоял из простых слов английского языка. Новый язык назвали «универсальным символическим кодом для начинающих» (Beginner All-Purpose Symbolic Instruction Code, или, сокращенно, BASIC). Годом рождения нового языка можно считать 1964. Сегодня универсальный язык Бейсик (имеющий множество версий) приобрел большую популярность и получил широкое распространение среди пользователей ПК различных категорий во всем мире. В значительно мере этому способствовало то, что Бейсик начали использовать как встроенный язык персональных компьютеров, широкое распространение которых началось в конце 70-х годов. Однако Бейсик неструктурный язык, и поэтому он плохо подходит для обучения качественному программированию. Справедливости ради следует заметить, что последние версии Бейсика для ПК (например, QBasic) стали более структурными и по своим изобразительным возможностям приближаются к таким языкам, как Паскаль.

Разработчики ориентировали языки на разные классы задач, в той или иной мере привязывали их к конкретной архитектуре ПК, реализовывали личные вкусы и идеи. В 60-е годы были предприняты попытки преодолеть эту «разноголосицу» путем создания универсального языка программирования. Первым детищем этого направления стал PL/1 (Programm Language One), разработанный фирмой IBM в 1967 году. Этот язык претендовал на возможность решать любые задачи: вычислительные, обработки текстов, накопления и поиска информации. Однако он

оказался слишком сложным, транслятор с него – недостаточно оптимальным и содержал ряд невыявленных ошибок. [1, 18]

Однако линия на универсализацию языков была поддержана. Старые языки были модернизированы в универсальные варианты: Алгол-68 (1968 г.), Фортран-77. Предполагалось, что подобные языки будут развиваться и усовершенствоваться, станут вытеснять все остальные. Однако ни одна из этих попыток не увенчалась успехом.

Язык ЛИСП появился в 1965 году. Основным в нем служит понятие рекурсивно определенных функций. Поскольку доказано, что любой алгоритм может быть описан с помощью некоторого набора рекурсивных функций, то ЛИСП по сути является универсальным языком. С его помощью ПК может моделировать достаточно сложные процессы, в частности – интеллектуальную деятельность людей. [5]

Пролог разработан во Франции в 1972 году для решения проблем «искусственного интеллекта». Пролог позволяет в формальном виде описывать различные утверждения, логику рассуждений и заставляет ПК давать ответы на заданные вопросы.

Значительным событием в истории языков программирования стало создание в 1971 году языка Паскаль. Его автор – швейцарский ученый Никлаус Вирт. Вирт назвал его в честь великого французского математика и религиозного философа XVII века Блеза Паскаля, который изобрел первое суммирующее устройство, именно поэтому новому языку было присвоено его имя. Этот язык первоначально разрабатывался как учебный язык структурного программирования, и, действительно, сейчас он является одним из основных языков обучения программированию в школах и вузах.

В 1975 году два события стали вехами в истории программирования – Билл Гейтс и Пол Аллен заявили о себе, разработав свою версию Бейсика, а Вирт и Йенсен выпустили классическое описание языка «Pascal User Manual and Report».

Не менее впечатляющей, в том числе и финансовой, удачи добился Филип Кан, француз, разработавший в 1983 году систему Турбо-Паскаль. Суть его идеи состояла в объединении последовательных этапов обработки программы – компиляции, редактирования связей, отладки и диагностики ошибок – в едином интерфейсе. Турбо-Паскаль – это не только язык и транслятор с него, но еще и операционная оболочка, позволяющая пользователю удобно работать на Паскале.

Этот язык вышел за рамки учебного предназначения и стал языком профессионального программирования с универсальными возможностями. В силу названных достоинств Паскаль стал источником многих современных языков программирования. С тех пор появилось несколько версий Турбо-Паскаля, последняя – седьмая. [4, 19]

Фирма Borland/Inprise завершила линию продуктов Турбо-Паскаль и перешла к выпуску системы визуальной разработки для Windows – Delphi.

Большой отпечаток на современное программирование наложил язык Си (первая версия – 1972 год), являющийся очень популярным в середе разработчиков систем программного обеспечения (включая операционные системы). Этот язык создавался как инструментальный язык для разработки операционных систем, трансляторов, баз данных и других системных и прикладных программ. Си сочетает в себе как черты языка высокого уровня, так и машинно-ориентированного языка, допуская программиста ко всем машинным ресурсам, чего не обеспечивают такие языки как Бейсик и Паскаль.

Период с конца 60-х до начала 80-х годов характеризуется бурным ростом числа различных языков программирования, сопровождавшим кризис программного обеспечения. В январе 1975 года Пентагон решил навести порядок в хаосе трансляторов и учредил комитет, которому было предписано разработать один универсальный язык. В мае 1979 года был объявлен победитель – группа ученых во главе с Жаном Ихбиа. Победивший язык окрестили Ада, в честь Огасты Ады Левлейс. Этот язык предназначен для создания и длительного (многолетнего) сопровождения больших программных систем, допускает возможность параллельной обработки, управления процессами в реальном времени. [1, 2, 11]

В течение многих лет программное обеспечение строилось на основе операционных и процедурных языков, таких как Фортран, Бейсик, Паскаль, Ада, Си. По мере эволюции языков программирования получили широкое распространение и другие, принципиально иные, подходы к созданию программ.

1.2 Низкоуровневые языки программирования

Язык низкого уровня — это язык программирования, который касается аппаратных компонентов, то есть работает на уровне компьютера. Он не имеет абстракции по отношению к компьютеру и работает для управления операционной семантикой

компьютера.

Язык низкого уровня также может называться как «родным» языком компьютера.

Низкоуровневые языки предназначены для работы и управления всей аппаратурой и инструкциями, заданными для архитектуры компьютера напрямую.

Низкоуровневые языки считаются ближе к компьютерам. Другими словами, их главной функцией является управление, разработка и взаимодействие с данными вычислительного оборудования и компонентов. Программы и приложения, написанные на низкоуровневом языке, непосредственно исполняются на вычислительном оборудовании без какой-либо интерпретации или перевода.

Язык машинного языка и язык ассемблера являются популярными примерами языков низкого уровня.

Машинный код — это единственный язык, который компьютер может обрабатывать напрямую без предварительного преобразования. В настоящее время программисты почти никогда не пишут программы непосредственно на машинном коде, потому что это требует внимания к многочисленным деталям, которые язык высокого уровня обрабатывает автоматически. Кроме того, он требует запоминания или поиска числовых кодов для каждой инструкции и его крайне трудно модифицировать. [4, 7, 9]

Истинный машинный код — это поток сырых, обычно двоичных, данных.

1.3 Высокоуровневые языки программирования

Языками программирования высокого уровня называют языки, которые ориентированы на человека, более близки и понятны ему. В таких языках не учитываются особенности компьютерных архитектур. Создаваемые программы на уровне исходных текстов средствами этих языков легко переносимы на другие платформы, имеющие транслятор соответствующего языка. Создание языков высокого уровня по времени совпадает с появлением языков программирования третьего поколения (60-е годы XX века). Это универсальные языки, с их помощью удается решать задачи из любых областей. Относительная простота, независимость от конкретного компьютера и возможность использования мощных синтаксических конструкций позволили резко повысить производительность труда программистов. К числу языков этого поколения относится Basic (Бейсик). Для

этого языка имеются и компиляторы, и интерпретаторы, а по популярности он занимает первое место в мире. Этот язык очень прост в изучении.

С начала 70-х годов XX века по настоящее время продолжается период языков четвертого поколения. Эти языки предназначены для реализации крупных проектов, повышения их надежности и скорости создания. Как правило, в эти языки встроены мощные операторы, позволяющие одной строкой описать такую функциональность, для реализации которой на языках младших поколений потребовались бы тысячи строк исходного кода. К числу языков этого поколения относятся: Pascal (Паскаль), C (Си), C++ (Си++), Java (Джава, Ява).

Паскаль во многом напоминает Алгол (язык 3-го поколения), но в нем ужесточен ряд требований к структуре программы и имеются возможности, позволяющие успешно применять его при создании крупных проектов.

Язык Си планировался для замены ассемблера, чтобы иметь возможность создавать столь же эффективные и компактные программы, и в то же время не зависеть от конкретного типа процессора. Этот язык во многом похож на Паскаль, однако имеет дополнительные средства (указатели) для прямой работы с памятью.

Си++ - это объектно-ориентированное расширение языка Си. Он имеет множество мощных возможностей, позволяющих резко повысить производительность труда программистов. Однако этот язык требует от программистов высокого уровня профессиональной подготовки. [15]

Язык Java был создан в начале 90-х годов XX века компанией Sun на основе Си++.

В этом языке исключены все низкоуровневые возможности языка Си++. Главной особенностью языка Java является то, что компиляция происходит не в машинный код, а в платформа-независимый байт-код (каждая команда занимает один байт). Этот байт-код может выполняться с помощью интерпретатора виртуальной Java-машины (Java Virtual Machine), версии которой сегодня созданы для любых платформ. Благодаря этому программы на Java можно переносить не только на уровне исходных текстов, но и на уровне двоичного байт-кода. Поэтому по популярности этот язык сегодня занимает второе место в мире после Бейсика. Особое внимание в развитии языка Java уделяется двум направлениям:

1) поддержке мобильных устройств и микрокомпьютеров, встраиваемых в бытовую технику (технология Jini);

2) созданию платформа-независимых программных модулей, способных работать на серверах в глобальных и локальных компьютерных сетях с различными операционными системами (технология Java Beans).

В середине 90-х годов XX века появились языки пятого поколения. Это языки объектно-ориентированного программирования. К ним относятся системы автоматического создания прикладных программ с помощью визуальных средств разработки. В этом случае становится необязательным знание основ программирования. Главная идея, которая заложена в основу этих языков, - возможность автоматического формирования результирующего текста на универсальных языках программирования. Этот результирующий текст программы потом необходимо откомпилировать. [12, 19]

В рамках данной главы были изучения исторические аспекты появления программирования, разница между низкоуровневыми и высокоуровневыми языками программирования.

2. Исследования высокоуровневых языков программирования

2.1 Язык C

Язык программирования C первоначально использовался для работы по разработке больших систем, особенно программ, которые составляют операционную систему. C был принят как язык разработки системы, поскольку он реализует программы, которые работают почти так же быстро, как код, написанный на ассемблере. C очень широко используется уже много лет. К некоторым областям в которых используется этот язык относятся следующие:

- операционные системы;

- компиляторы языка;

- текстовые редакторы;

- сетевые драйверы;

- базы данных;

- языковые переводчики.

Язык С составляет подмножества языка С++. Практически более новый язык включает в себя весь С и добавляет ему новые функции, а самое главное — объектно-ориентированную парадигму.

Программа на С может состоять от 3 до миллионов строк и может быть записана в один или несколько текстовых файлов с расширением «.c», например, hello.c. [7]

Разработка С первоначально не была целью его создателей. Фактически, различные обстоятельства и проблемы создали идеальную ситуацию для ее появления. В 1960-х годах Деннис Ритчи, который был сотрудником Bell Labs (AT&T), вместе с некоторыми из его коллег, работал над разработкой операционной системы, которая могла бы использоваться многими пользователями одновременно. Эта операционная система была известна как Multics, и она должна была позволить многим пользователям совместно использовать общие вычислительные ресурсы. Multics добавляли много преимуществ, но также имели много проблем. Это была большая система, и казалось, с точки зрения выгоды, что затраты перевешивают вложенные усилия и ресурсы. Постепенно Bell Labs отказались от проекта.

Программа на С состоит из различных, так называемых, токенов, а токен — это либо ключевое слово, идентификатор, константа, строковый литерал, либо символ. Например, следующий оператор С состоит из пяти токенов:

printf("Hello, World! \n");

Индивидуальные токены:

printf

(

"Hello, World! \n"

)

;

В программе на С точка с запятой является идентификатором утверждения. То есть каждое отдельное утверждение должно заканчиваться точкой с запятой. Он указывает конец одного логического объекта.

Ниже приводятся два разных утверждения:

printf("Hello, World! \n");

return 0;

Комментарии являются неким текстом помощи к программе на языке C, и они игнорируются компилятором. Они начинаются с “/ *” и заканчиваются символами “* /”, как показано ниже:

/* my first program in C */

Невозможно комментировать комментарии, и они не встречаются в строковых или символьных литералах.

Идентификатор на C — это имя, используемое для идентификации переменной, функции или любого другого определяемого пользователем элемента. Идентификатор начинается с буквы от A до Z, от a до z или символа подчеркивания '_', за которым следуют ноль или несколько букв, знаков подчеркивания и цифр (от 0 до 9). [10]

C не допускает знаков препинания, таких как “@, $ и %” внутри идентификаторов. C — язык программирования с учетом регистра. Таким образом.

Есть список наименований показывает зарезервированные слова в C. Эти зарезервированные слова не могут использоваться в качестве констант или переменных, или любых других имен идентификаторов.

Строка, содержащая только пробелы, возможно с комментарием, называется пустой строкой, и компилятор C полностью игнорирует ее.

Пробел — это термин, используемый в C для описания пробелов, вкладок, символов новой строки и комментариев. Пробел отделяет одну часть инструкции от другой и позволяет компилятору определить, где начинается один элемент в инструкции, такой как int, и следующий элемент. Поэтому в следующем утверждении: “int age;” должен быть по крайней мере один пробельный символ (обычно пробел) между int и age, чтобы компилятор мог их отличить. С другой стороны, в следующем утверждении: “fruit = apples + oranges; // get the total fruit” никакие пробельные символы не нужны между фруктами и =, или между = и яблоками, хотя можно включать некоторые, если нужно увеличить удобочитаемость.

Типы данных в C относятся к обширной системе, используемой для объявления переменных или функций разных типов. Тип переменной определяет, сколько места занимает он в хранилище и как интерпретируется шаблон бита. [8]

Типы в C можно классифицировать следующим образом:

1. Основные типы.

Они являются арифметическими типами и далее подразделяются на: (a) целые типы и (b) типы с плавающей точкой.

2. Перечисляемые типы.

Они снова являются арифметическими типами, и они используются для определения переменных, которые могут назначать только определенные дискретные целочисленные значения во всей программе.

3. Тип void.

Указатель типа void указывает, что значение не доступно.

4. Производные типы.

Они включают (a) типы указателей, (b) типы массивов, (c) типы структуры, (d) типы Союза и (e) типы функций.

Типы массивов и типы структуры в совокупности относятся к смешанным типам. Тип функции указывает тип возвращаемого значения функции [16].

Тип void указывает, что значение не доступно. Он используется в трех типах ситуаций:

1. Функция возвращается как “ничего”.

В C есть различные функции, которые не возвращают никакого значения, или можно сказать, что они возвращают пустоту. Функция без возвращаемого значения имеет тип возврата как void. Например, void exit (int status).

2. Аргументы функции как недействительные.

В C есть различные функции, которые не принимают никаких параметров. Функция без параметра может принимать “пустоту”. Например, int rand (void);

3. Указатели на “пустоту”.

Указатель типа void * представляет адрес объекта, но не его тип. Например, функция выделения памяти void * malloc (size_t size); возвращает указатель на void, который может быть передан любому типу данных.

Переменная — это не что иное, как название области хранения, которой могут манипулировать программы. Каждая переменная в C имеет определенный тип, который определяет размер и расположение памяти переменной; диапазон значений, которые могут быть сохранены в этой памяти; и набор операций, которые могут быть применены к переменной. [6]

Имя переменной может состоять из букв, цифр и символа подчеркивания. Он должен начинаться с буквы или подчеркивания. Буквы верхнего и нижнего регистра различны, потому что C чувствителен к регистру.

Язык программирования C также позволяет определять различные, другие типы переменных, таких как «Перечисление», «Указатель», «Массив», «Структура» и т.д.

Определение переменной сообщает компилятору, где и сколько хранилища следует создать для переменной. Определение переменной указывает тип данных и содержит список одной или нескольких переменных этого типа следующим образом:

type variable_list;

Здесь тип должен быть допустимым типом данных C, включая char, w_char, int, float, double, bool или любой пользовательский объект; и переменная_list может состоять из одного или нескольких имен идентификаторов, разделенных запятыми. Некоторые действительные объявления показаны ниже:

int i, j, k;

char c, ch;

float f, salary;

Значения int i, j, k; объявляет и определяет переменные i, j и k; которые инструктируют компилятор создавать переменные с именем i, j и k типа int. [4, 15]

Переменные могут быть инициализированы (назначено начальное значение) в их объявлении. Инициализатор состоит из знака равенства, за которым следует

постоянное выражение следующим образом:

type variable_name = value;

Некоторые примеры:

extern int d = 3, f = 5; // declaration of d and f.

int d = 3, f = 5; // definition and initializing d and f.

byte z = 22; // definition and initializes z.

char x = 'x'; // the variable x has the value 'x'.

Для определения без инициализатора переменные со статической продолжительностью хранения неявно инициализируются с помощью NULL (все байты имеют значение 0); начальное значение всех других переменных не определено.

Объявление переменной предоставляет гарантию компилятору, что существует переменная с заданным типом и именем, чтобы компилятор мог продолжить компиляцию, не требуя полной информации о переменной. Определение переменной имеет смысл только во время компиляции, компилятор нуждается в определении фактической переменной во время связывания программы.

Объявление переменной полезно, когда используется несколько файлов, и нужно определить свою переменную в одном из файлов, которые будут доступны во время компоновки программы. Будет использоваться ключевое слово extern для объявления переменной в любом месте. Хотя можно объявить переменную несколько раз в своей программе на C, ее можно определить только один раз в файле, функции или блоке кода.

Существует два вида выражений в C

l. Lvalue.

Выражения, относящиеся к ячейке памяти, называются выражениями lvalue. Значение l может отображаться как в левой, так и в правой части задания.

2. Rvalue.

Термин rvalue относится к значению данных, которое хранится на некотором адресе в памяти. Rvalue — это выражение, которое не может иметь назначенное ему значение, что означает, что rvalue может отображаться с правой стороны, но не с левой стороны задания. [8]

Переменные являются lvalues, и поэтому они могут появляться в левой части задания. Числовые литералы являются значениями r и поэтому они не могут быть назначены и не могут отображаться с левой стороны.

2.2 Java и его экосистема

Java является популярным языком программирования веб-приложений и имеет ряд преимуществ.

Одним из основных преимуществ Java в разработке программного обеспечения и приложений является «то, что это кроссплатформенный инструмент. Благодаря виртуальной машине JVM (Java Virtual Machine) среда выполнения Java может переводить код в машинный код, совместимый с собственной операционной системой, будь то Windows, iOS или Linux. Эта универсальность, и в особенности кроссплатформенная функциональность, сразу же делает ее мощным инструментом для крупных организаций, занимающихся разработкой программного обеспечения. [2]

Java является языком программирования на основе C, который позволяет разработчикам легко его изучить, если они работали с другими языками на основе C.

Наиболее часто на сервере Java используется в связке с фреймворком Spring.

Структура Spring поддерживает большинство функциональных возможностей инфраструктуры приложений Enterprise. Ниже приведены некоторые основные преимущества Spring Framework.

1. Spring позволяет разработчикам разрабатывать корпоративные приложения, используя POJO (обычный старый Java-объект). Преимущество разработки приложений с использованием POJO заключается в том, что не нужно иметь корпоративный контейнер, такой как сервер приложений, но есть возможность использовать надежный контейнер сервлетов. [13]

2. Spring упрощает процесс разработки, используя абстракцию существующих технологий, таких как сервлеты, jsps, jdbc, jndi, rmi, jms и Java mail и т.д.
3. Spring поставляется с некоторыми из существующих технологий, таких как ORM-инфраструктура, структура ведения журнала, J2EE и JDK-таймеры и т. Д. Поэтому не нужно явно интегрировать эти технологии.
4. Spring WEB framework имеет хорошо продуманную веб-структуру MVC, которая обеспечивает отличный альтернативный интерфейс веб-фреймворка.
5. Spring может исключить создание одноэлементных и заводских классов.
6. Spring обеспечивает последовательный интерфейс управления транзакциями, который может уменьшаться до локальной транзакции и масштабироваться до глобальных транзакций (с использованием JTA).
7. Spring Framework включает поддержку управления бизнес-объектами и предоставление их услуг компонентам уровня представления, так что веб-приложения и настольные приложения могут обращаться к тем же объектам.
8. Spring Framework использует наилучшую практику, которая была доказана на протяжении многих лет в нескольких приложениях и формализована как шаблоны проектирования.
9. Приложение Spring может использоваться для разработки приложений различного типа, таких как автономные приложения, автономные графические приложения, веб-приложения и апплеты.
10. Spring поддерживает как xml, так и анотационные конфигурации.
11. Spring Framework позволяет разрабатывать автономную, настольную, двухшинную архитектуру и распределенные приложения.
12. Spring предоставляет встроенные средства промежуточного программного обеспечения, такие как объединение пулов, управление транзакциями и т.д.
13. Spring обеспечивает легкость, которая может быть активирована без использования веб-сервера или сервера приложений.
14. Более того, Spring Framework состоит из функций, организованных примерно в 20 модулей, эти модули сгруппированы в Core Container, Data Access/Integration, Web, AOP (аспектно-ориентированное программирование), Instrumentation and Test. [10]

2.3 PHP

Причиной популярности PHP является его несколько преимуществ. PHP наиболее подходит для веб-разработки.

К его основным достоинствам относятся:

1. Кроссплатформенность.

Все приложения на базе PHP могут работать на разных типах платформ. Поддержка PHP поддерживается большинством операционных систем, некоторые из которых включают Solaris, UNIX, Windows и Linux. Указанные платформы могут использоваться для написания кодов на PHP, а также просмотра веб-страниц или запуска приложений на базе PHP.

PHP легко взаимодействует с MySQL и Apache. Легкая интеграция PHP может быть выполнена с использованием различных других технологий, таких как Java, и главное нет необходимости в повторном использовании. Поэтому, экономия времени и денег, придаёт ему большое преимущество.

2. Простое подключение к базе данных.

Язык программирования, такой как PHP, широко используется в Интернете, что требует очень частого подключения к базе данных. Таким образом, наличие функции, которая может помочь PHP легко подключаться к базе данных, является обязательной. Некоторые веб-сайты, такие как веб-сайты электронной торговли, требуют хорошей системы управления базами данных.

PHP имеет встроенный модуль, который помогает легко подключаться к базе данных. Поэтому PHP пользуется большим спросом в области веб-разработки, где должен быть разработан веб-сайт, ориентированный на данные. PHP значительно сокращает время, необходимое для разработки веб-приложения, которое нуждается в эффективной системе управления базами данных. [11]

3. Легко использовать.

PHP широко используется, потому что он прост в использовании. В отличие от других языков программирования, которые являются сложными, PHP прост, свободен, чист и организован, поэтому он является удобным для новых пользователей. PHP имеет хорошо организованный синтаксис, который является логичным одновременно удобным.

PHP не требует интенсивного изучения или руководства для его использования. Командные функции PHP легко понятны, так как пользователь может легко понять из названия самих команд, что он делает. Человек, который только начал изучать PHP, уже может легко программировать, потому что синтаксис несколько похож на

С, поэтому проще создавать и оптимизировать приложение с помощью PHP.

Скорость - это основная потребность в веб-разработке. Есть люди, которые сталкиваются с проблемой медленного подключения к Интернету и медленной скоростью передачи данных. Кроме того, сайты с быстрой загрузкой всегда предпочитают люди по всему миру. По сравнению с другими языками программирования, PHP является самым быстрым языком программирования.

В обычных условиях для подключения к базе данных требуется много времени, когда пользователь пытаетесь получить определенные данные из базы данных. Это занимает много времени при подключении к базе данных, а затем выполнение инструкции и, наконец, получение данных. PHP выполняет этот набор задач быстрее, чем другие языки сценариев. PHP быстрее работает как при подключении к базе данных, так и при использовании других важных приложений. [11]

Высокая скорость работы PHP дает ему преимущество перед другими языками сценариев и дает ему приоритет в решении важных задачах, таких как администрирование сервера и функции почты.

4. Открытый исходный код.

Одним из важных преимуществ PHP является то, что это Open Source проект. Поэтому PHP легко доступен и полностью свободен. В отличие от других языков сценариев, используемых для веб-разработки, которые требуют от пользователя оплаты файлов поддержки, PHP открыт для всех, в любое время и в любом месте.

Новичок в PHP не должен беспокоиться о поддержке, поскольку PHP поддерживается и развивается большой группой разработчиков PHP, которая помогает создавать сообщество поддержки PHP, которое помогает людям в реализации и манипулировании PHP.

Но есть у него и недостатки. Каждый язык программирования имеет свой собственный набор преимуществ и недостатков. Аналогично, PHP также имеет свой собственный набор недостатков, к таковым относятся следующие:

1. Безопасность.

Веб-сайты должны быть максимально безопасными, чтобы владелец веб-сайта и пользователи веб-сайта были в безопасности и защищены. PHP доступен абсолютно бесплатно, т.е. имеет открытый исходный код. С одной стороны, это преимущество PHP и, с другой стороны, создает угрозу для веб-сайта, разработанного с

использованием PHP.

2. Плохая обработка ошибок.

Разработчики считают, что у PHP плохое качество обработки ошибок. PHP не хватает средств отладки, необходимых для поиска ошибок и предупреждений. У PHP меньше средств отладки по сравнению с другими языками программирования.

2.4 Objective-C

Objective-C — основной язык программирования, который используется при написании программного обеспечения для OS X и iOS. Это надмножество языка программирования C, который добавляет в него объектно-ориентированные возможности и динамический вариант выполнения. Objective-C наследует синтаксис, примитивные типы и операторы управления потоками C и добавляет синтаксис для определения классов и методов. Он также добавляет поддержку уровня языка для управления графами объектов и объектных литералов, обеспечивая динамическую типизацию и привязку, откладывая многие преобразования до выполнения.

Несмотря на то, что кластерные классы отделены от языка, их использование сильно накладывается на кодирование с помощью Objective-C, и многие языковые функции зависят от поведения, предлагаемого этими классами. [15]

При создании приложений для OS X или iOS большую часть времени программист работает с объектами. Эти объекты являются экземплярами классов Objective-C, некоторые из которых предоставлены, а некоторые из них нужно писать самому.

При написании собственного класса, нужно начать с составления описания класса, который детализирует предполагаемый публичный интерфейс для экземпляров класса. Этот интерфейс включает общедоступные свойства для инкапсуляции релевантных данных, а также список методов. Объявления метода указывают сообщения, которые может получить объект, и включают информацию о параметрах, которые требуются всякий раз, когда вызывается метод. Разработчик также предоставит реализацию класса, которая включает исполняемый код для каждого метода, объявленного в интерфейсе.

Вместо того, чтобы создавать совершенно новый класс для предоставления дополнительных возможностей над существующим классом, можно определить

категорию для добавления пользовательского поведения в существующий класс. Разработчик может использовать категорию для добавления методов в любой класс, включая классы, для которых нет исходного кода реализации.

Если есть исходный код для класса, можно использовать расширение класса для добавления новых свойств или изменения атрибутов существующих свойств. Расширения классов обычно используются, чтобы скрыть личное поведение для использования либо в одном файле исходного кода, либо в рамках частной реализации пользовательской структуры. [18]

Большая часть работы в приложении Objective-C происходит в результате того, что объекты отправляют сообщения друг другу. Часто эти сообщения определяются методами, явно указанными в интерфейсе класса. Иногда, однако, полезно иметь возможность определять набор связанных методов, которые не привязаны непосредственно к определенному классу.

Objective-C использует протоколы для определения группы связанных методов, таких как методы, которые объект может вызвать для своего делегата, которые являются необязательнымы. Любой класс может указать, что он принимает протокол, а это значит, что он также должен обеспечивать реализацию всех необходимых методов в протоколе.

Ценности и коллекции часто представлены как объекты Objective-C.

В Objective-C обычно используется класс Cocoa или Cocoa Touch для представления значений. Если NSString класс используется для строк символов, то NSNumber класс для различных типов чисел, таких как целое или с плавающей точкой, а также NSValue класс для других значений, таких как структуры C. Можно использовать любой из примитивных типов, определенных языком C, например, int, float или char. [15]

Коллекции обычно представлены как экземпляры одного из классов коллекций, такие как NSArray, NSSet, или NSDictionary, каждый из которых используются для сбора других объектов Objective-C.

Блоки — это языковая функция, введенная для C, Objective-C и C ++ для представления единицы работы; они инкапсулируют блок кода вместе с захваченным состоянием, что делает их похожими на замыкания из других языков программирования. Блоки часто используются для упрощения общих задач, таких как перечисление, сортировка и тестирование коллекции. Они также позволяют

легко планировать задачи для параллельного или асинхронного выполнения с использованием таких технологий, как Grand Central Dispatch (GCD).

Хотя Objective-C включает в себя синтаксис обработки исключений, исключения для использования Cocoa и Cocoa Touch используются только для новых ошибок в ходе программирования (например, из-за отсутствия доступа к границам), которые должны быть исправлены до того, как приложение будет отправлено.

Все другие ошибки, в том числе проблемы времени выполнения, такие как нехватка места на диске или невозможность доступа к веб-службе, представлены экземплярами NSError класса. Приложение должно планировать ошибки и решать, как лучше их обрабатывать, чтобы представить наилучший пользовательский опыт, когда что-то пойдет не так.

Код цели C соответствует установленным соглашениям

При написании кода Objective-C нужно иметь в виду ряд установленных правил кодирования. Имена методов, например, начинаются с строчной буквы и используют верблюжью нотацию для нескольких слов; например, doSomething или doSomethingElse. Однако важна не только капитализация; нужно убедиться, что код максимально читабелен, что означает, что имена методов должны быть выразительными, но не слишком подробными. [18]

Кроме того, существует несколько соглашений, которые необходимы, иметь ввиду вы нужно воспользоваться языковыми или функциональными возможностями. Например, методы доступа к ресурсам должны следовать строгим соглашениям об именах, чтобы работать с такими технологиями, как Key-Value Coding (KVC) или Key-Value Observing (KVO).

2.5 Ruby

Rails — это платформа разработки веб-приложений, написанная на языке программирования Ruby. Он призван облегчить программирование веб-приложений, сделав ставку на то, что каждый разработчик должен суметь легко начать на нём писать. Это позволяет писать меньше кода при сравнении с другим большим количества языков и фреймворков.

Rails — это надежное программное обеспечение. [7]

Одна из ключевых характеристик Ruby заключается в том, что это объектно-ориентированный язык программирования. В отличие от некоторых языков, которые функционируют, предоставляя компьютеру список выполняемых задач, объектно-ориентированный язык программирования позволяет разработчику создавать виртуальные объекты в своем коде. Каждый объект может иметь свои собственные специфические атрибуты и способности и может взаимодействовать с другими объектами для выполнения действий. Затем группам объектов могут быть даны инструкции на основе этих атрибутов и то, как они соотносятся друг с другом. Если думать об объектах, подобных виртуальным строительным блокам, группу блоков A можно использовать для построения башни, а группе блоков B можно дать указание создать шаги, ведущие к вершине башни. Объектно-ориентированный язык делает разработку менее сложной, предоставляя структурам программистов работать с тем, что напоминает здание в реальном мире. Объекты также могут быть легко использованы в будущих программах.

Философия Rails включает в себя два основных руководящих принципа:

- «не повторяйся» или DRY - это принцип разработки программного обеспечения, в котором говорится, что «у каждой части знаний должно быть одно, недвусмысленное, авторитетное представление внутри системы». Не повторяя одну и ту же информацию снова и снова, код более удобен в обслуживании, более расширяемый и менее подвержен ошибкам;
- «конец конфигурации» — Rails имеет мнение о наилучшем способе делать много вещей в веб-приложении и по умолчанию использует этот набор соглашений, вместо того, чтобы требовать, чтобы указывались всякие мелочи через бесконечные файлы конфигурации. [7]

Другими словами, веб-инфраструктура Rails не только сокращает время повторного кодирования повторяющихся задач, но с помощью Rails-кода разработчики сохраняют свою общую работу более чистой, менее подверженной ошибкам (плохой, неэффективный код), и легче устраняют проблемы, когда те возникают.

Таким образом, мы знаем, что Rails — это платформа, которая позволяет Rails-разработчикам использовать язык Ruby для разработки веб-сайтов.

В рамках данной главы были рассмотрены основные высокоуровневые языки программирования.

ЗАКЛЮЧЕНИЕ

В рамках работы решены следующие задачи:

- изучена соответствующая литература;
- изучена история появления «программирования»;
- проанализированы низкоуровневые и высокоуровневые языки программирования;
- проведён анализ высокоуровневых языков программирования;
- сделать выводы по проделанной работе.

Важное значение для развития высокоуровневых языков программирования имела разработка во второй половине 1950-х годов трех языков – Fortran, COBOL, Lisp. Философия, стоящая за этими языками, заключается в создании высокоуровневой системы обозначений, облегчающей программисту написание программ.

Новой тенденцией является появление языков программирования немного более высокого уровня. Такого рода языки характеризуются наличием дополнительных структур и объектов, ориентированных на прикладное использование. Прикладные объекты, в свою очередь, требуют минимальной настройки в виде параметров и моментально готовы к использованию. Использование ультра-высокоуровневых языков программирования снижает временные затраты на разработку программного обеспечения и повышает качество конечного продукта за счет, опять-таки, уменьшения объёма исходных кодов.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Вирт Н. Построение компиляторов. — ДМК Пресс, 2013. — 188 с.
2. Владстон Феррейра Фило Теоретический минимум по Computer Science. Все что нужно программисту и разработчику. — Питер, 2018. — 224 с.
3. Дьюхерст С. C++. Священные знания. 2-е издание, исправленное. — Символ-Плюс, 2012. — 230 с.
4. Клеменс Б. Язык C в XXI веке. — ДМК Пресс, 2015. — 378 с.
5. Мартин Р. Чистый код: создание, анализ и рефакторинг. — Питер, 2018. — 464 с.
6. Нахавандипур В. iOS. Приемы программирования. — Питер, 2014. — 1000 с.

7. Майкл Хартл Ruby on Rails для начинающих. Изучаем разработку веб-приложений на основе Rails. — ДМК Пресс, 2017. — 574 с.
8. Орлов С. А. Теория и практика языков программирования. — Питер, 2017. — 688 с.
9. Филимонова Е. Информационные технологии в профессиональной деятельности. — КноРус, 2017. — 483 с.
10. Родли Джон Создание Java-апплетов.- The Coriolis Group,Inc.,1996, Издательство НИПФ "ДиаСофт Лтд.",1996
11. Робин Никсон Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. — Питер, 2016. — 767 с.
12. Тарасов С. СУБД для программиста. Базы данных изнутри. — СОЛОН-Пресс, 2015. — 322 с.
13. Эферган Майкл Java: справочник.- QUE Corporation, 2012, Издательство "Питер Ком", 2012
14. Jeffrey A. Hoffer Modern Database Management. — Pearson, 2015. — 601 p.
15. Aaron Hillegass, Mikey Ward Objective-C Programming: The Big Nerd Ranch Guide (2nd Edition). — Big Nerd Ranch Guides, 2013. — 325 p.
16. K. N. King C Programming: A Modern Approach, 2nd Edition. — W. W. Norton & Company, 2008. — 832 p.
17. Stephen G. Kochan Programming in C (4th Edition). — Addison-Wesley Professional, 2014. — 544 p.
18. Stephen G. Kochan Programming in Objective-C. — Addison-Wesley Professional, 2013. — 552 p.
19. John R. Vacca Computer and Information Security Handbook. — Morgan Kaufmann, 2017. — 1280 p.
20. Regina O. Obe , Leo S. Hsu PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database 3rd Edition. — O'Reilly Media, 2017. — 314 p.
21. Laine Campbell Database Reliability Engineering: Designing and Operating Resilient Database Systems. — O'Reilly Media, 2017. — 294 p.