Содержание:

Введение

По мере усиления зависимости общества от вычислительных систем надежность и гибкость последних приобретает все большее значение. Последствиями этой зависимости являются растущие требования к качеству программного обеспечения, возможностям его быстрой разработки, масштабируемости, способности к расширению и совместимости с современными аппаратными средствами. Современный уровень развития компьютерных технологий предъявляет повышенные требования к квалификации специалистов всех отраслей, в том числе к уровню владения информационными технологиями. Поскольку деятельность современного производства и бизнеса в целом тесно связано со своевременностью получения информации, избыточностью и точностью этой информации, а также способностью эффективно ее использовать. Для свободной ориентации в информационных потоках современный специалист любого профиля должен уметь получать, обрабатывать и использовать информацию, прежде всего, с помощью компьютерной техники, сетей и других новейших средств связи, в том числе и уметь, обращаться с языками программирования. Используемый в конкретном проекте язык программирования в значительной степени определяет скорость разработки и реализации, простоту сопровождения, возможность переноса, создаваемого в рамках этого проекта программного обеспечения.

В ответ на возрастание роли программного обеспечения и требований к нему, специалистами были разработаны различные языки программирования, обеспечивающие повышение эффективности, мобильности, надежности и упрощение сопровождения, создаваемого с их помощью программного обеспечения. Появление большого числа современных языков программирования существенно усложнило выбор того или иного языка программирования для решения конкретной задачи или набора задач. И хотя появилось достаточно большое число работ, посвященных сравнению языков программирования и их оценки с точки зрения совместимости, производительности и функциональности, в этих работах практически не было уделено внимание вопросу выбора языка программирования исходя из требований к разрабатываемому продукту.

В этой курсовой работе изложена техника создания методологии сравнения языков программирования и сред разработки. Данная работа построена на базе перечня вопросов, лежащих в основе сравнения и оценки языков программирования. Этот перечень вопросов был сформирован, исходя из посылки его последующего применения для сравнения и оценки современных языков программирования, для примера использованы языки Бейсик, Си, Паскаль. С каждым из включенных в этот перечень вопросов ассоциирован ряд конкретных подразделов, ответы на которые необходимы для формирования характеристики некоторого языка программирования. Кроме того, каждый из включенных в этот перечень вопросов сопровождается дополнительной информацией, представляющей собой критерии, которые могут быть использованы для оценки языка программирования на основе уже сформированной его характеристики.

1. Языки программирования

Язык программирования — это система обозначений, служащая для точного описания программ или алгоритмов для вычислительной техники. Языки программирования являются искусственными языками. От естественных языков они отличаются ограниченным числом “слов” и очень строгими правилами записи команд (операторов). Поэтому при применении их по назначению они не допускают свободного толкования выражений, характерного для естественного языка.

Можно сформулировать ряд требований к языкам программирования и классифицировать языки по их особенностям.

Базовые требования, предъявляемые к языкам программирования:

наглядность - использование в языке по возможности уже существующих символов, хорошо известных и понятных как программистам, так и пользователям ЭВМ;

единство - использование одних и тех же символов для обозначения одних и тех же или родственных понятий в разных частях алгоритма. Количество этих символов должно быть по возможности минимальным;

гибкость - возможность относительно удобного, несложного описания распространенных приемов математических вычислений с помощью имеющегося в языке ограниченного набора изобразительных средств;

модульность - возможность описания сложных алгоритмов в виде совокупности простых модулей, которые могут быть составлены отдельно и использованы в различных сложных алгоритмах;

однозначность - недвусмысленность записи любого алгоритма. Отсутствие ее может привести к неверным ответам при решении задач [2].

В настоящее время в мире существует несколько сотен реально используемых языков программирования. Для каждого есть своя область применения.

Любой алгоритм, есть последовательность предписаний, выполнив которые можно за конечное число шагов перейти от исходных данных к результату. В зависимости от степени детализации предписаний обычно определяется уровень языка программирования — чем меньше детализация, тем выше уровень языка.

1.1. История развития языков программирования

Программа – алгоритм, записанный на языке программирования. Программа – последовательность операторов языка. Языки программирования – искусственные языки, строго формализованные; существует правила записи операторов языка – синтаксис языка.

1. Машинный язык (40-50 годы XX в.).

Программы на машинном языке – очень длинные последовательности единиц и нулей (машинный код), являлись машинно-зависимыми, т.е. для каждой вычислительной системы необходимо составлять свою программу.

2. Ассемблер (начало 50-ых годов XX в.).

Вместо 1 и 0 используются операторы (MOV, ADD, SUB и т.д.), которые похожи на английские слова. Программы на ассемблере также являются машинно-зависимыми. Для преобразования в машинный код использовался компилятор (спец. программа – переводчик в машинный код).

3. Первые языки программирования высокого уровня.

С середины 50-ых гг. XX в. начали создавать первые языки программирования высокого уровня. Эти языки были Машино-независимыми (не привязаны к определенному типу вычислительной техники). Но для каждого языка были

разработаны собственные компиляторы.

Примеры таких языков: FORTRAN (FORmulaTRANslator; 1954) предназначен для научных и технических расчетов; COBOL (1959) был предназначен в основном для коммерческих приложений (обрабатывал большие объемы нечисловых данных) – CommonBusiness-OrientedLanguage); язык BASIC (Beginner’sAllPurposeInstuctionCode – универсальный язык символьных инструкций для начинающих) (1964 г.)

4. Алгоритмические языки программирования.

С начала 80-ых г. XX в. начали создаваться языки программирования, которые позволили перейти к структурному программированию (использование операторов ветвления, выбора, цикла и практически отказ от частого использования операторов перехода (goto). К этим языкам относятся: язык Pascal (назван его создателем Никлаусом Виртом в честь великого физика Блеза Паскаля; 1970); язык Си, позволяющий быстро и эффективно создавать программный код (1971)

5. Языки объектно-ориентированного программирования

(90-ые г. XX в.). В основу этих языков положены программные объекты, которые объединяют данные и методы их обработки. В этих языках сохранялся алгоритмический стиль программирования. Для них были разработаны интегрированные среды программирования, позволяющие визуально конструировать графический интерфейс приложений:

язык C++ (1983) - продолжение алгоритмического языка Си;

язык ObjectPascal (1989) был создан на основе языка Pascal. После создания среды программирования – Delphi (1995);

язык VisualBasic(1991) был создан корпорацией Microsoft на основе языка Qbasic (1975) для разработки приложений с графическим интерфейсом в среде ОС Windows.

6. Языки программирования для компьютерных сетей.

В 90-ые годы XX в. в связи с бурным развитием Интернета были созданы языки, обеспечивающие межплатформенную совместимость. На подключенных к Интернету компьютерах с различными ОС (Windows, Linux, MacOS и др.) могли выполняться одни и те же программы. Исходная программа компилируется в промежуточный код, который исполняется на компьютере встроенной в браузер

виртуальной машиной:

язык Java - объектно-ориентированный язык был разработан фирмой SunMicrosystems для создания сетевого программного обеспечения (1995);

язык JavaScript - язык сценариев Web-страниц (компания Netscape). (1995)

7. Языки программирования на платформе .NET.

Интегрированная среда программирования VisualStudio .Net, разработанная корпорацией Microsoft, позволяет создавать приложения на различных языках объектно-ориентированного программирования, в том числе:

На языке Visual Basic .Net (на основе Visual Basic) - 2003 г.;

на языке VisualC# (C-шарп) – на основе языков C++ и J – 2003 г.;

на языке VisualJ# (J-шарп) – на основе Java и JavaScript – 2003 г.

Интерпретаторы и компиляторы

Для того, чтобы процессор имел возможность выполнить программу, программа и данные должны быть загружены в оперативную память. Необходимо, чтобы в ОП была размещена программа - транслятор, автоматически переводящий с языка программирования в машинные коды. Трансляторы бывают двух типов: интерпретаторы и компиляторы. Интерпретатор – программа, которая обеспечивает последовательный перевод операторов программы с одновременным их выполнением. Достоинством интерпретатора является удобство отладки (поиск ошибок), недостаток – сравнительно малая скорость выполнения. Компилятор переводит весь текст программы на машинный язык и сохраняет его в исполняемом (бинарном) файле (обычно с расширением .exe, для семейства операционных систем Windows).

Системы объектно-ориентированного программирования содержат программу-транслятор и позволяют работать в режиме как интерпретатора, так и компилятора. На этапе разработки и отладки проекта используется режим интерпретатора, а для получения готового приложения – режим компилятора [6].

1.2. Классификация языков программирования

Уровень языка программирования определяет близость языка к естественному, человеческому языку, чем выше уровень, тем ближе. По этому критерию можно выделить следующие типы языков программирования:

· машинные;

· машинно-ориентированные (ассемблеры);

· машинно-независимые (языки высокого уровня).

Машинные и машинно-ориентированные языки — это языки низкого уровня, требующие указания мелких деталей процесса обработки данных. Разные типы процессоров имеют разные наборы команд. Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется языком программирования низкого уровня. В данном случае “низкий уровень” не значит “плохой”. Имеется в виду, что операторы языка близки к машинному коду и ориентированы на конкретные команды процессора. [2]

При программировании на машинном языке программист может держать под своим контролем каждую команду и каждую ячейку памяти, использовать все возможности имеющихся машинных операций. Но процесс написания программы на машинном языке очень трудоемкий и утомительный. Программа получается громоздкой, труднообозримой, ее трудно отлаживать, изменять и развивать.

Поэтому в случае, когда нужно иметь эффективную программу, в максимальной степени учитывающую специфику конкретного компьютера, вместо машинных языков используют близкие к ним машинно-ориентированные языки (ассемблеры).

Язык ассемблера — это машинно-зависимый язык низкого уровня, в котором короткие мнемонические имена соответствуют отдельным машинным командам. Используется для представления в удобочитаемой форме программ, записанных в машинном коде. [2]

Язык ассемблера позволяет программисту пользоваться текстовыми мнемоническими (то есть легко запоминаемыми человеком, похожими на слова естественного языка) кодами, по своему усмотрению присваивать символические имена регистрам компьютера и памяти, а также задавать удобные для себя способы адресации. Кроме того, он позволяет использовать различные системы счисления (например, десятичную или шестнадцатеричную) для представления числовых констант, использовать в программе комментарии и др.

С помощью языков низкого уровня создаются очень эффективные и компактные программы, так как разработчик получает доступ ко всем возможностям аппаратной платформы. С другой стороны, при этом требуется очень хорошо понимать устройство компьютерной техники, затрудняется отладка больших приложений, а окончательная программа не может быть перенесена на компьютер с другим типом (архитектурой) процессора. Подобные языки обычно применяют для написания небольших системных приложений, драйверов устройств, модулей стыковки с нестандартным оборудованием, когда важнейшими требованиями становятся компактность, быстродействие и возможность прямого доступа к аппаратным ресурсам. В некоторых областях, например в компьютерной графике, на языках низкого уровня пишутся библиотеки, эффективно реализующие алгоритмы обработки изображений и трехмерной графики, требующие интенсивных вычислений.

Таким образом, программы, написанные на языке ассемблера, требуют значительно меньшего объема памяти и времени выполнения. Знание программистом языка ассемблера и машинного кода дает ему понимание архитектуры вычислительного комплекса. Несмотря на то, что большинство специалистов в области программного обеспечения разрабатывают программы на языках высокого уровня, наиболее мощное и эффективное программное обеспечение полностью или частично написано на языке ассемблера.

Языки высокого уровня - были разработаны для того, чтобы освободить разработчика программы от учета технических особенностей конкретных вычислительных комплексов или платформ, их архитектуры. Языки программирования высокого уровня имитируют естественные языки, используя некоторые слова разговорного языка и общепринятые математические символы. Эти языки более удобны для понимания человеком. Уровень языка характеризуется степенью его близости к естественному, человеческому языку. Машинный язык не похож на человеческий, он крайне беден в своих изобразительных средствах. Средства записи программ на языках высокого уровня более выразительны и привычны для человека. Например, алгоритм вычисления по сложной формуле не разбивается на отдельные операции, а записывается компактно в виде одного выражения с использованием привычной математической символики. Составить свою или понять чужую программу на таком языке гораздо проще.

Важным преимуществом языков высокого уровня является их универсальность, независимость от аппаратной платформы. Программа, написанная на таком языке, может выполняться на разных машинах. Составителю программы не нужно знать

систему команд вычислительной машины, на которой он предполагает проводить обработку данных. При переходе на другую вычислительную систему программа не требует переделки. Такие языки – не только средство общения человека с машиной, но и людей между собой. Программа, написанная на языке высокого уровня, легко может быть понята любым специалистом, который знает язык и характер реализованной задачи.

Таким образом, можно сформулировать основные преимущества языков высокого уровня перед машинными:

алфавит языка высокого уровня значительно шире алфавита машинного языка, что существенно повышает наглядность текста программы;

набор операций, допустимых для использования, не зависит от набора машинных операций, а выбирается из соображений удобства формулирования алгоритмов решения задач определенного класса;

формат предложений достаточно гибок и удобен для использования, что позволяет с помощью одного предложения задать достаточно содержательный этап обработки данных;

необходимые операции задаются с помощью общепринятых математических обозначений;

данным в языках высокого уровня присваиваются индивидуальные имена, выбираемые разработчиком программы;

в языке может быть предусмотрен значительно более широкий набор различных типов данных по сравнению с набором машинных типов данных.

Таким образом, языки высокого уровня в значительной мере являются машинно-независимыми. Они облегчают работу программиста и повышают надежность создаваемых программ.

Основные компоненты алгоритмического языка:

· алфавит,

· синтаксис,

· семантика.

Алфавит — это фиксированный для данного языка набор основных символов, т.е. "букв алфавита", из которых должен состоять любой текст на этом языке — никакие другие символы в тексте не допускаются.

Синтаксис — это правила построения фраз, позволяющие определить, правильно или неправильно написана та или иная фраза. Точнее говоря, синтаксис языка представляет собой набор правил, устанавливающих, какие комбинации символов являются осмысленными предложениями на этом языке.

Семантика определяет смысловое значение предложений языка. Являясь системой правил истолкования отдельных языковых конструкций (выражений), семантика устанавливает, какие последовательности действий описываются теми или иными фразами языка и, в конечном итоге, какой алгоритм определен данным текстом программы на алгоритмическом языке.

Языки высокого уровня делятся на:

· процедурные;

· логические;

· объектно-ориентированные.

Процедурные языки предназначены для однозначного описания алгоритмов. При решении задачи процедурные языки требуют в той или иной форме явно записать процедуру ее решения.

Первым шагом в развитии процедурных языков программирования было появление проблемно-ориентированных языков. В этом названии нашел отражение тот факт, что при их разработке идут не от «машины», а «от проблемы» (от задачи): в таком языке программирования стремятся максимально полно учесть специфику класса задач, для решения которых его предполагается использовать. Например, для многих научно-технических задач характерны большие расчеты с использованием сложных формул, поэтому в ориентированных на такие задачи языках вводят удобные средства их записи. Использование понятий, терминов, символов, привычных для специалистов соответствующей области знаний, облегчает им изучение языка, упрощает процесс создания и отладки программного продукта.

Разнообразие классов задач привело к тому, что на сегодняшний день разработано несколько сотен алгоритмических языков. Однако, широкое распространение и международное признание получили лишь полтора десятка языков. Среди них в

первую очередь следует отметить: Fortran и Algol - языки, предназначенные для решения научно-технических задач, Cobol – для решения экономических задач, Basic – для решения небольших вычислительных задач в диалоговом режиме. В принципе каждый из этих языков можно использовать для решения задач не своего класса. Однако, как правило, применение оказывается не удобным.

В то же время в середине 60-х годов начали разрабатывать алгоритмические языки широкой ориентации – универсальные языки. Обычно они строились по принципу объединения возможностей узко-ориентированных языков. Среди них наиболее известны PL/1, Pascal, C, C++, Modula, Ada. Однако, как любое универсальное средство, такие широко-ориентированные языки во многих конкретных случаях оказываются менее эффективными [2].

Логические языки- (Prolog, Lisp, Mercury, KLO и др.) ориентированы не на запись алгоритма решения задачи, а на систематическое и формализованное описание задачи с тем, чтобы решение следовало из составленного описания. В этих языках указывается что дано и что требуется получить. При этом поиск решения задачи возлагается непосредственно на ЭВМ.[3]

Объектно-ориентированные языки (Object Pascal, C++, Java, Objective C и др.). Основная идея объектно-ориентированных языков заключается в стремлении связать данные с обрабатывающими эти данные процедурами в единое целое - объект.

Объектно-ориентированный подход использует следующие базовые понятия:

· объект;

· свойство объекта;

· метод обработки;

· событие;

· класс объектов.

Объект — совокупность свойств (параметров) определенных сущностей и методов их обработки (программных средств).

Свойство — это характеристика объекта и его параметров. Все объекты наделены определенными свойствами, совокупность которых выделяют (определяют) объект.

Метод — это набор действий над объектом или его свойствами.

Событие — это характеристика изменения состояния объекта.

Класс — это совокупность объектов, характеризующихся единообразием применяемых к ним методов обработки или свойств.

Существуют различные объектно-ориентированные технологии, которые обеспечивают выполнение важнейших принципов объектного подхода:

инкапсуляция;

наследование.

Под инкапсуляцией понимается скрытие полей объекта с целью обеспечения доступа к ним только посредством методов класса (т. е. скрытие деталей, несущественных для использования данного объекта). Инкапсуляция (объединение) означает сочетание данных и алгоритмов их обработки, в результате чего и данные, и процедуры во многом теряют самостоятельное значение.

Класс может иметь образованные от него подклассы. При построении подклассов осуществляется наследование данных и методов обработки объектов исходного класса. [2]

Фактически объектно-ориентированное программирование можно рассматривать как модульное программирование нового уровня, когда вместо во многом случайного, механического объединения процедур и данных акцент делается на их смысловую связь.

Программа на объектно-ориентированном языке, решая некоторую задачу, по сути, описывает часть мира, относящуюся к этой задаче. Описание действительности в форме системы взаимодействующих объектов естественнее, чем в форме взаимодействующих процедур. [2]

2. Обзор современных языков программирования

Алгоритмические языки программирования представляют собой один из способов записи алгоритма. Язык программирования является строго формализованным, то есть все команды записываются по определенным правилам и отступления от этих

правил будут определятся как ошибка в коде. Например, в русском языке можно при разделении элементов перечисления поставить запятую (,) или точку с запятой (;). А в языке программирования при записи команд нельзя изменить ни одного знака - возникает ошибка.

Правила записи команд на конкретном языке называются синтаксисом языка. Синтаксис определяет, какая команда будет считаться правильной, а какая нет. Например, в языке Basic команды CLS и FORI=1 TO 10 считаются правильными, а команды CLERSCREEN и FORIFROM 1 TO 10 - неправильными.

Каждая команда, записанная на языке программирования, имеет определенное значение, то есть заставляет компьютер выполнять определенные действия с данными или аппаратными ресурсами. Правила, определяющие смысл команд, называются семантикой языка. Например, команда CLS вызывает очистку экрана.

Каждый язык имеет алфавит – набор символов, которые можно использовать при записи программ на этом языке. Разные версии одного и того же языка могут немного различаться алфавитом.

Программа, написанная на языке программирования, состоит из команд (операторов), задающих последовательность действий. Эти действия выполняются над некоторыми объектами. Объектами могут быть числа, текстовые строки, переменные и другие объекты. Языки отличаются друг от друга множеством допустимых объектов и набором операций, которые можно выполнять над этими объектами. [7]

Программа, написанная на языке программирования, представляет собой машинописный текст. Чтобы компьютер мог выполнять команды, содержащиеся в этой программе, надо перевести программу в набор понятных компьютеру инструкций, записанных в двоичной форме (машинный код). Такой перевод называется трансляцией.

По способу трансляции языки делятся на:

· компиляторы

· интерпретаторы

В компиляторах перевод всего текста программы в код осуществляется сразу, и создаются исполняемый файл, который затем можно неоднократно запускать.

В интерпретаторах при запуске программы каждая ее строчка последовательно переводится в код и выполняется; затем переводится в код и выполняется другая строчка, и так далее.

Каждый оператор языка представляет собой мнемоническое (условное) обозначение машинной команды или набора команд. Естественно, что каждый тип процессора имеет свой набор команд, а значит, свой ассемблер. Интерпретаторы и компиляторы решают эту проблему совместимости. Ассемблеры же в чистом виде используются сегодня для создания драйверов, программирования различных устройств, а также для написания фрагментов программ, где очень важно время выполнения, так как на ассемблере можно написать максимально эффективную программу [5].

Современные языки программирования делят на процедурно-ориентированные и объектно-ориентированные, но в настоящее время граница между этими видами стерлась. Эти языки используются чаще всего для решения самых разнообразных задач. И хотя каждый из языков имеет свои особенности, что делает его наиболее эффективными для решения определенного вида задач, но в принципе для решения любой задачи можно выбирать любой язык программирования. Например, язык Lisp используется для создания экспертных систем. Язык Java используется для разработки сетевых (Web)- приложений.

Процесс создания программы включает несколько этапов. Раньше для реализации каждого этапа использовались специальные средства. Например, текст программы сначала набирался в текстовом редакторе. Затем с помощью специальной команды запускался транслятор, чтоб перевести текст программы в машинный код. Затем другой командой запускался компоновщик, чтобы объединить вновь написанную программу с разработанными ранее фрагментами и создать исполняемый файл. Наконец, программа запускалась, и тут обнаруживалось, что результаты получаются совсем не такие, как надо. Для поиска ошибок использовался отладчик, который позволял, например, посмотреть промежуточные результаты каких-то вычислений. После того, как ошибки были найдены, приходилось исправлять их в текстовом редакторе и начинать весь процесс сначала. Таким образом, разработка и отладка программы была долгим и трудоемким делом.

В настоящее время существуют средства, позволяющие выполнять все действия в рамках единой среды. Поэтому сейчас чаще говорят не о языках программирования, а об интегрированных средствах разработки.

Интегрированная среда разработки обычно включает в себя:

- текстовый редактор – для набора текста программы;
- компилятор (или интерпретатор) - для перевода программы в машинный код;
- компоновщик - для объединения при необходимости нескольких программ “запускатель программ”, который позволяет выполнить разрабатываемую программу, не выходя из среды разработки;
- отладчик, который позволяет посмотреть промежуточные результаты, сделать паузу в заданной строке программы, либо при изменении значения заданной переменной;
- справочную систему, описывающую особенности конкретной реализации языка.

Для одного и того же языка могут существовать разные среды разработки. Например, для языка С есть среда TurboC и BorlandC [7].

Среди универсальных языков программирования в настоящее время наиболее распространены:

2.1. Си его разновидности

Си (англ. C) — компилируемый статически типизированный язык программирования общего назначения, разработанный в 1969—1973 годах сотрудником Bell Labs Деннисом Ритчи как развитие языка Би. Первоначально был разработан для реализации операционной системы UNIX, но впоследствии был перенесён на множество других платформ. Согласно дизайну языка, его конструкции близко сопоставляются типичным машинным инструкциям, благодаря чему он нашёл применение в проектах, для которых был свойственен язык ассемблера, в том числе как в операционных системах, так и в различном прикладном программном обеспечении для множества устройств — от суперкомпьютеров до встраиваемых систем. Язык программирования Си оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для таких языков программирования, как C++, C#, Java и Objective-C.

Си++ [C++] - Язык программирования высокого уровня, созданный Бьярном Страустрапом на базе языка Си. Является его расширенной версией, реализующей принципы объектно-ориентированного программирования. Используется для

создания сложных программ. Для IBM PC наиболее популярной является система Turbo C++ фирмы Borland (США).

C# (C Sharp) – “Си Шарп”: объектно-ориентированный язык программирования, о разработке которого в 2000 г. объявила фирма Microsoft. По своему характеру он напоминает языки C++ и Java и предназначен для разработчиков программ, использующих языки C и C++ для того, чтобы они могли более эффективно создавать Интернет-приложения. Указывается, что C# будет тесно интегрирован с языком XML [1].

2.2. Паскаль

Паскаль [PASCAL - акроним с французского – Program Appliqueala Selectionetla Compilation Automatique de la Litterature] - Процедурно-ориентированный язык программирования высокого уровня, разработанный в конце 1960-х гг. Никлаусом Виртом, первоначально для обучения программированию в университетах. Назван в честь французского математика XVII века Блеза Паскаля.

В своей начальной версии Паскаль имел довольно ограниченные возможности, поскольку предназначался для учебных целей, однако последующие его доработки позволили сделать его хорошим универсальным языком, широко используемым в том числе для написания больших и сложных программ. Существует ряд версий языка (например, ETH Pascal, USD Pascal, Turbo Pascal) и систем программирования на этом языке для разных типов ЭВМ. Для IBM PC наиболее популярной является система Turbo Pascal фирмы Borland (США).

Delphi является «наследником» языка Паскаль; основные операторы в этих языках одинаковы. Но Delphi имеет средство для работы с различными графическими объектами (создания форм, кнопок, меню), а также для обработки сложных структур данных. Поэтому он очень популярен при разработке различных Windows-приложений [1]. По настоящее время широко используется в учебных целях как первый язык программирования, это связано с тем, что его простой синтаксис и очень близкая к естественному языку семантика позволяют максимально снизить порог вхождения для начинающих изучать программирование.

2.3. Бейсик

Бейсик [BASIC - Beginner’sAll-purposeSymbolicInstructionCode] - Язык программирования высокого уровня, разработанный в 1963 - 1964 гг. в Дартмутском колледже Томасом Куртом и Джоном Кемени.

Первоначально предназначался для обучения программированию. Отличается простотой, легко усваивается начинающими программистами благодаря наличию упрощенных конструкций языка Фортран и встроенных математических функций, алгоритмов и операторов. Существует множество различных версий Бейсика, которые не полностью совместимы друг с другом. Некоторые реализации Бейсика включают средства обработки данных и наборов данных.

Большинство версий Бейсика используют интерпретатор, который преобразует его компоненты в машинный код и позволяет запускать программы без промежуточной трансляции. Некоторые более совершенные версии Бейсика позволяют использовать для этой цели трансляторы. На IBM PC широко используются Quick Basic фирмы Microsoft, Turbo Basic фирмы Borland и Power Basic (усовершенствованная версия Turbo Basic, распространяемая фирмой Spectra Publishing). В начале 1999 г. фирма Microsoft выпустила версию языка Visual Basic 6.0 (VB 6.0), предназначенного для создания многокомпонентных программных приложений для систем уровня предприятий [1].

3. Выбор инструментов разработки

При упоминании выбора инструментов реализации проекта чаще всего подразумевают выбор языка программирования и среды проекта, при этом сравнивают программные продукты и поддерживаемые этими средами языки программирования и инструменты разработки. Использование возможностей средств разработки приложений позволяет автоматизировать процесс создания продукта. Инструментальные средства позволяют:

-создавать интерфейс, используя стандартные компоненты;

-передавать управление процессам, в зависимости от состояния системы;

-создавать оболочки для баз данных, как и сами базы данных;

-разрабатывать более надежные программы путем обработки исключительных ситуаций, возникающих при некорректной работе программы.

Современные средства разработки характеризуются параметрами:

-поддержка объектно-ориентированного стиля программирования;

-возможность использования CASE-технологий, как для проектирования разрабатываемой системы, так и для разработки моделей реляционных баз данных;

-использование визуальных компонент для наглядного проектирования интерфейса;

-поддержка БД.

Для выбора технологии, языка и среды программирования проводится теоретический анализ, опираясь на источники литературы и интернет доступных современных инструментальных средств для разработки приложений по следующим критериям:

- название, версия, фирма производитель, под управлением каких ОС функционирует (зависимость от платформы);

- подход к разработке программного обеспечения (структурный, объектно-ориентированный)

- механизмы доступа к БД (например ADO.NET, LINQ, IBX [4]);

- утилиты для работы с БД;

- поддержка стандарта языка SQL;

- наличие компонент для работы с БД (невизуальные и визуальные компоненты);

- наличие компонент построения отчетов и диаграмм;

- поддержка графического интерфейса пользователя (Таких как оконный интерфейс Windows);

- средства поддержки транзакций (параллельная работа нескольких пользователей с БД);

- простота/сложность работы с инструментальным средством;

- возможность создания запускаемого файла.

3.1. Выбор языка разработки

Язык программирования должен отвечать всем современным требованиям и соответствовать тем задачам, которые встают перед программистом сегодня.

Сегодня предоставляемое программисту многообразие возможностей позволяет, начиная новый проект любой сложности, выбрать язык программирования, наиболее подходящий для его реализации. При выборе языка программирования необходимо учитывать требования проекта к скорости работы, потребляемым ресурсам, необходимости кроссплатформенного использования, а также сроки реализации проекта. Кроме того, для некоторых проектов может стать существенным фактор популярности языка программирования. Для популярного языка имеется, как правило, большое количество учебной и справочной литературы, кроме того, использованный язык программирования может стать решающим фактором при выборе программы конечным пользователем. Не менее важно при выборе языка программирования учитывать опыт других разработчиков при решении похожих задач, это поможет избежать типичной ошибки – «изобретение колеса».

Рассмотрев основные концепции современных языков программирования можно выделить несколько наиболее актуальных задач, стоящих сегодня перед программистами: Проектирование межплатформенных приложений – не зависящих от используемой платформы. Разработка Интернет–приложений – работающих через Интернет, но не поддерживаемых браузером непосредственно. Создание Web-приложений – клиент-серверных архитектур. Создание мобильных приложений, в том числе клиент-серверной архитектуры. Возможность быстрого создания приложений и средств автоматизации. Создание приложений для нового, но уже достаточно популярного направления IoT (интернета вещей), в котором аппаратные средства, традиционно программируемые на низкоуровневых языках, программируются на современных объектно-ориентированных языках, открывая мир умных устройств для начинающих специалистов. Также актуальны приложения полностью и эффективно использующие все ресурсы современных платформ.

При выборе языка программирования нужно учитывать множество факторов. Например, если при разработке динамической Web-страницы вы в качестве наилучшего варианта выберите JavaServer Pages (JSP)/сервлеты, другие могут

предпочесть PHP, Python или другой аналогичный язык сценариев. Не существует какого-то одного языка, который является наилучшим выбором. Можно отдать предпочтение определенным факторам, таким как производительность и безопасность корпоративных приложений, по сравнению с другими факторами, такими как количество строк кода. Немаловажную роль играет опыт разработчика в применении конкретного языка программирования. Умением разработчика решать конкретную задачу средствами определенного языка. Большое влияние оказывает общепринятые в среде разработчиков стандарты на принадлежность определенных языков для реализации конкретных задач. Любое решение сопряжено с какими-то компромиссами.

После получения проекта или задания нужно выполнить подготовительную работу до решения поставленной задачи. Зачастую выбор языка не рассматривается как часть этой подготовительной работы.

При выборе языка для персонального проекта можно положиться на свои личные предпочтения. Здесь может оказаться важным количество строк кода; очевидным выбором будет язык, позволяющий выполнить задачу при помощи 10 строк кода вместо 20. Сначала хочется получить решение, а потом позаботиться об удобочитаемости или производительности.

В проектах для команд разработки применяется другой сценарий. Для решения конкретной проблемы группы разработчиков создают компоненты, взаимодействующие и взаимосвязанные между собой. На выбор языка могут повлиять такие факторы, как переносимость программы на другую платформу или доступность ресурсов.

Правильный выбор языка программирования поможет создать компактное, простое в отладке, расширении, документировании и исправлении ошибок решение. При выборе языка программирования учитываются следующие факторы:

Целевая платформа;

Гибкость языка;

Время исполнения проекта;

Производительность;

Поддержка и сообщество.

Самым важным фактором является платформа, на которой программа будет работать. Рассмотрим для примера Java™ и C. Если программа написана на C и должна работать на машинах с Windows® и Linux®, потребуются компиляторы для платформ и два разных исполняемых файла. В случае с Java сгенерированного байт-кода будет достаточно для выполнения программы на любом компьютере, на котором установлена виртуальная Java-машина.

Аналогичный аргумент применим и для Web-сайтов. Они должны выглядеть и работать одинаково во всех браузерах. Использование стилей CSS3 и языка разметки HTML5 без проверки совместимости с браузерами приведет к разному отображению и поведению сайта в разных браузерах.

Гибкость языка определяется тем, насколько легко можно добавлять к существующей программе новые функциональные возможности. Это может быть добавление нового набора функций или использование сторонней библиотеки для добавления нового функционала. При этом рассматривают следующие вопросы, связанные с гибкостью:

Можно ли использовать новый функционал без подключения новой библиотеки?

Если нет, доступна ли эта функциональность в библиотеке языка?

Если эта функциональность не встроена в язык и не доступна в библиотеке, какие усилия нужно приложить для ее создания с нуля?

До принятия решения необходимо знать, как спроектирована программа, и какие функциональные возможности оставлены на потом.

Хотя сравнение этих языков технически некорректно, рассмотрим Perl и Python. Perl имеет встроенную поддержку регулярных выражений. В Python необходимо импортировать дополнительный модуль из стандартной библиотеки.

Время исполнения – это время, необходимое для создания рабочей версии программы, т.е. версии, готовой для работы в производственных условиях и выполняющей предусмотренные функции. При расчете этого времени необходимо учитывать не только логику управления, но и логику представления.

Время исполнения проекта в значительной зависит от объема кода. Теоретически, чем легче изучить язык и чем меньше объем кода, тем меньше это время.

Например, сайт управления контентом на PHP-сценариях можно разработать за несколько дней, в то время как создание кода сервлетов может занять несколько месяцев, при условии, что вы начали изучать оба языка с нуля.

Каждая программа и платформа имеет определенный предел производительности, и на эту производительность влияет используемый при разработке язык. Существует множество способов сравнения скорости работы в одинаковой среде программ, написанных на разных языках. Можно использовать различные эталонные тесты, хотя их результаты не являются конкретной оценкой производительности какого бы то ни было языка.

Рассмотрим два варианта Web-приложения, написанных на Java и на Python. На основании данных тестирования можно прийти к заключению, что в одинаковой среде приложение, написанное на Java, должно работать быстрее, чем приложение, написанное на Python. А как насчет самой среды? Если средой является персональный компьютер или физический сервер, это справедливо, поскольку вычислительная мощность ограничена. А если взять Web-приложение, работающее в облачном сервисе, то такое приложение может использовать практически неограниченную процессорную мощность, и обе программы возвратят результаты почти за одно и то же время. Теперь основным фактором выбора будет количество строк кода и удобство обслуживания.

Производительность языка нужно учитывать в случае, когда целевая среда не предлагает широкой масштабируемости, – например, при разработке для мобильных устройств или аппаратных контроллеров.

Язык программирования, как и хорошая программа, должен опираться на твердую поддержку сообщества. Язык с активным сообществом разработчиков скорее всего будет популярнее замечательного языка, помощь по которому трудно найти.

Поддержка сообщества – это вики-сайты, форумы, учебные руководства и готовые дополнительные библиотеки, развивающие язык. Современные разработчики редко работают автономно, для решения типовых задач все чаще используются готовые решения в виде подключаемых библиотек, что сокращает время на изучение документации в поисках подсказки в написании алгоритма, выполняющего требующуюся для реализации алгоритма операцию с данными. Если у языка много сторонников, это увеличивает шансы того, что ранее кто-нибудь сталкивался с аналогичной задачей и уже написал об этом на вики-сайте или на форуме, написал код реализующий данный функционал. [8]

3.2. Выбор среды разработки программ

Среда разработки является основным инструментом создания информационной системы или программного продукта ради которого был выбран язык программирования, платформа, создана команда, проведена серия брифингов с заказчиком и командой разработки, разработано и согласованно с заказчиком техническое задание на создание программного продукта, в связи с чем необходимо так же тщательно подойти к выбору среды, в которой данный программный продукт или группа продуктов будет создана.

При выборе среды разработки информационной системы нужно придерживаться нескольким моментам:

-среда должна являться средой визуальной разработки (т.е. в процессе создания программы будет видно, как она будет выглядеть во время выполнения);

-должна быть реализована поддержка объектно-ориентированного стиля программирования (например, наследование, полиморфизм);

-должна быть реализована поддержка процедурного программирования (т.е. что каждая программа состоит из набора процедур, каждая из которых решает свою задачу, при этом одна процедура может вызывать любую другую);

-невысокие требования среды и создаваемой программы к ресурсам аппаратных средств, если такие ограничения присутствуют в конечной среде применения создаваемого ПО;

-должна быть предусмотрена возможность использования стандартных компонентов при создании интерфейса;

-должно быть предусмотрено наличие библиотек для работы с графикой;

-программа должна быть создана с использованием визуальных компонентов для интерфейса;

-разрабатываемый программный продукт должен быть надёжным (должна быть предусмотрена обработка исключительных ситуаций, возникающих при неправильной работе программы).

У каждого программиста есть свой взгляд на модель разработки, определяющийся его прошлым опытом, степенью освоения тех или иных инструментальных средств.

Любая современная среда разработки позволяет производить настройку и адаптацию под те или иные требования: изменение интерфейса, режимов работы, назначения горячих клавиш, установка дополнительных средств («плагинов») и т.п. В арсенале каждого опытного программиста должны быть свои приемы разработки, собственные вспомогательные средства. Он имеет собственные вкусы и предпочтения. Используя настройки, программист может сделать работу в среде более удобной для себя и, тем самым, более эффективной.

Создатели инструментальных средств закладывают, как правило, избыточный набор возможностей и программисты практически никогда не используют инструментальное средство на все 100%.

Обычно среда разработки ПО предназначена для разработки только на одном языке программирования. А такая среда разработки как интегрированная, предоставляет право выбрать создателю программы язык программирования для разработки, удобный разработчику (из поддерживаемых данной средой языков). Примером тому служат: Visual Studio, Komodo, Kylix, NetBeans, Eclipse.

Данные среды разработки программного обеспечения (или подобные) можно осуществлять весь цикл разработки программного обеспечения. Но есть также интегрированные среды, которые предназначены для одного программного языка. Например такая среда как Visual Basic, являющаяся частью программного комплекса Office от компании Microsoft.

Для больших (или командных) проектов в среду разработки должны быть включены файловый менеджер, интегрированная среда разработки программного обеспечения, Система Управления БД (включающая инструмент отчётов таких как Crystal Reports Windows Forms Viewer и Crystal Reports Engine [4]), система отчетов о процессе разработки, журнал изменений для контроля версий продукта, а также интеграция с внешними системами контроля версий и система планирования разработки представляющая собой техническое задание на разработку разложенное на временную шкалу, разделенное по блокам на участников команды и определяющую последовательность создания и взаимосвязь модулей (частей) будущего программного продукта.

Интегрированные среды разработки удобны для командных проектов, постольку в таких средах можно производить весь цикл создания программного обеспечения.

Подводя итоги нужно сказать о том, что интегрированные среды разработки ПО позволяют программистам сократить время на написание приложений, снизить затратность на написание (разработку), повысить удобность разработки — что и является одной из основных целей программной инженерии.

Выводы

Сегодняшний ритм жизни, предъявляет достаточно высокие требования к скорости создания, внедрения и поддержки новых продуктов и технологий, развитию и обновлению существующих технологий и инструментов. Неотъемлемой частью современных технологий и устройств является программное обеспечение обеспечивающее функционирование данных продуктов. Эффективность данного программного обеспечения во многом зависит от правильности выбора среды разработки, технологии и языка программирования использованного для реализации проекта.

Технология программирования во многом определяется языком программирования, на котором пишутся программы. В языке могут быть заложены средства, влияющие на технологичность и архитектуру разрабатываемой системы (например, объектно-ориентированность, модульность и т.п.). Обычно выбирают ту модель разработки и те языки программирования, которые хорошо знают члены коллектива разработчиков. Выбирать новую технологию, которую предстоит осваивать в процессе разработки – риск провалить проект.

Для успешной реализации проекта необходимо изучить требования к конечному продукту, его массовость, специфичность, область применения, ответственность, безопасность, все это повлияет на выбор языка программирования, а следовательно, на выбор команды и среды разработки продукта.

При выборе языка программирования стоит опираться на несколько факторов. Немаловажным из них являются навыки команды разработки, если она уже сформирована или продукт создается устоявшимся коллективом, как внутренний проект или делается для внешнего заказчика, организацией занимающейся разработкой программного обеспечения. Вторым весомым фактором являются требования к безопасности и отказоустойчивости приложения, для специфических устройств, устройств повышенной опасности, медицинских аппаратов, контроллеров устройств входящих в состав центров обработки материалов высокой точности, военной и космической промышленности больше подходят низкоуровневые языки, их узкая применимость и совместимость с небольшим

количеством устройств, играет в данном случае положительную роль, повышая эффективность использования ресурсов и безопасность. Это увеличивает срок разработки ПО, предъявляет высокие требования к знаниям команды в предметной области, но обеспечивает необходимую надежность. В случае же когда приложение выполняет роль средства обмена информации, передачи и обработки данных, в том числе не требующих высокой точности результатов, становится возможным использование высокоуровневых языков. Дополнительными факторами в выборе языка программирования могут быть требования обеспечения совместимости со сторонним программным обеспечением, необходимость в наличии интерфейсов взаимодействия с внешними устройствами, другим программным обеспечением и требование функционирования разрабатываемого ПО на определенных аппаратных платформах, сложность возлагаемых на ПО вычислений, специфичность данных передаваемых на обработку приложению и необходимость соединения с внешними базами данных для получения данных и сохранения результатов обработки данных. Немаловажную роль в выборе языка программирования играет область применения ПО, современные языки программирования зачастую ориентированы на создание ПО определенного типа, имеют встроенные средства для реализации конкретных задач. Часть из них предназначены для создания WEB-приложений, не взаимодействующих с аппаратной частью вычислительных систем, это может в свою очередь обеспечивать дополнительную безопасность данных. К таким языкам можно отнести PHP, Python, JavaScript. В противовес им существуют языки, на которых удобнее разрабатывать программное обеспечение, используемое непосредственно на рабочих станциях в качестве прикладных программ, или ПО микроконтроллеров. В данном случае мы говорим о языках C и его потомках, Pascal, Basic и других подобных. На стыке этих областей можно выделить отдельным пунктом язык программирования Java и его виртуальную машину, как средство взаимодействия онлайн интерфейсов, сетевого оборудования, мобильных устройств и аппаратного обеспечения устройств, на которых выполняется данное ПО и прочего программного обеспечения, в том числе с WEB-приложениями. Данный язык очень часто применяется для проверки подлинности данных и идентификации пользователя со стороны клиентской машины в банковских программных продуктах и системах обмена защищенной информацией или информацией, требующей подтверждения «электронной подписью».

В свою очередь выбор среды разработки напрямую зависит от выбора языка разработки ПО, чаще всего для языка программирования существует несколько сред разработки, бесплатных коммерческих и условно бесплатных. Так же

отличиями различных сред могут быть наличие определенных функций, упрощающих разработку ПО, таких как контроль синтаксиса, наличие средств отладки, различные удобства в виде подсветки кода, наличие средств контроля версий и возможности совместной разработки ПО группой разработчиков. Все эти функции так же могут стать частью списка требований к будущей среде разработки и помогут сделать правильный выбор.

Существуют так же обратные специфические ситуации, когда выбор среды стоит на первом месте перед выбором языка разработки, такое происходит, когда набор всего необходимого функционала, необходимого для реализации конкретного проекта, присутствует только в одной среде разработки, в таком случае выбор языка определяется автоматически если данная среда предназначена для разработки на конкретном языке, либо выбор ведется из поддерживаемых средой языков в случае, если среда является универсальной. К таким средам относятся Visual Studio, NetBeans, IntelliJ IDEA. Дополнительным фактором в выборе среды является конечная платформа для которой создается ПО, так как на одних и тех же языках программирования создаются программные продукты для различных операционных систем, аппаратных платформ и мобильных устройств, а среды разработки, обеспечивающие максимальную совместимость с конечными платформами, могут быть разными.

Таким образом можно сделать заключение о том, что выбор языка программирования, а также среды разработки, требует внимательного изучения требований к конечному продукту, таких его параметров, как надежность, совместимость с аппаратными средствами, сторонним программным обеспечением, наличие возможности к масштабированию, возможность переноса на другие платформы, необходимость в дальнейшем расширении функционала ПО. Так же следует обратить внимание на избыточность средств конкретного языка и среды разработки для решения поставленной задачи, выбирать максимальный по функциям и возможностям инструмент не всегда целесообразно в случаях решения задач низкой и средней сложности, ввиду отсутствия необходимости в большинстве возможностей предоставляемых такой средой или языком, так как изучение сложного инструмента и сложность его использования для разработки может отнимать большую часть времени процесса разработки, поиска ошибок и отладки, значительно увеличивая срок сдачи проекта. Исходя из составленного, на основе данных требований, чек-листа возможно провести сравнительный анализ и выбрать максимально подходящий для реализации проекта язык программирования и среду разработки.

Список литературы

1. C ++, TurboPasckal,QBasik: Эволюция языков программирования http://langprog.far.ru/historylangprog.html. -27.05.10.

2. Информатика/Курносов А.П., Кулев С.А., Улезько А.В. и др.; Под ред. А.П. Курносова.-М.: КолосС, 2005.-272 с

3. Макарова Н.В. Информатика /под ред. Проф. Н.В. Макаровой. — М.: Финансы и статистика, 1997. — 768 с.: ил.

4. Малышев Р.А. Локальные вычислительные сети: Учебное пособие/ РГАТА. – Рыбинск, 2005. – 83 с.

5. Островский В.А. Информатика: учеб. для вузов. М.: Высшая школа, 2000. —511 с.: ил.

6. Семакин И.А., Информатика: Базовый курс /Семакин И.А., Залогова Л., Русаков С., Шестакова Л. – Москва: БИНОМ.,2005. – 105с.

7. Симонович С.В. Информатика. Базовый курс/Симонович С.В. и др. — СПб.: издательство "Питер", 2000. — 640 с.: ил.

8. Jerry Reghunadh and Neha Jain Selecting the optimal programming language. Factors to consider Published on September 13, 2011 https://www.ibm.com/developerworks/web/library/wa-optimal/index.html?S_TACT=105AGX99&S_CMP=CP