Содержание:

ВВЕДЕНИЕ

На сегодняшний день, и на достигнутом этапе освоения компьютерных и информационных технологий, очень сложно вообразить по-настоящему квалифицированного специалиста, который не владел бы информационными технологиями хотя бы на начальном уровне. Это связано, помимо прочего с тем, что деятельность любого сотрудника в той или иной степени зависит от умения изучать, добывать и обрабатывать информацию, а на сегодняшний день – это нужно делать еще и оперативно.

Учитывая современный напор и количество нефильтрованной информации, поступающей пользователям ежедневно, появляется ряд требований, которые необходимо выполнить сегодняшнему специалисту. Но основным из всех является умение свободного поиска, обработки, распределения использования информации. Естественно, что среди таких требований также находится необходимость хотя бы первоначального знакомства с языками программирования.

Актуальность выбранной темы также обуславливается тем, что прогресс компьютерных технологий определяет процесс появления новых разнообразных знаковых систем для записи алгоритмов – языков программирования.

Объектом исследования послужила классификация языков программирования и критерии выбора среды и языка разработки.

Исходя из вышесказанного, определяется цель исследования: изучение классификации языков программирования, анализ критериев выбора среды и языка программирования.

Для достижения данной цели необходимо выполнить следующие задачи:

1. рассмотреть классификацию языков программирования;
2. изучить критерии выбора среды разработки;
3. проанализировать критерии выбора языка разработки.

Глава 1. Классификация языков

1.1 Базовая иерархия языков

Программирование – это искусство создавать программные продукты, которые написаны на языке программирования[1]. Язык программирования – это формальная знаковая система, которая предназначена для написания программ, понятной для исполнителя (в нашем рассмотрении – это компьютер).

Язык программирования (англ. Programming language) – система обозначений для описания алгоритмов и структур данных, определенная искусственная формальная система, средствами которой можно выражать алгоритмы. Язык программирования определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполняет исполнитель (компьютер) под ее управлением[2].

Со времени создания первых программируемых машин было создано более двух с половиной тысяч языков программирования. Ежегодно их число пополняется новыми. Некоторыми языками умеет пользоваться только небольшое число их собственных разработчиков, другие становятся известны миллионам людей. Профессиональные программисты обычно применяют в своей работе несколько языков программирования.

Языки программирования низкого уровня

Первым компьютерам приходилось программировать двоичными машинными кодами. Однако программировать таким образом – достаточно трудоемкая и сложная задача. Для упрощения этой задачи стали появляться языки программирования низкого уровня, которые позволяли задавать машинные команды в более понятном для человека виде. Для преобразования их в двоичный код были созданы специальные программы - трансляторы.

Пример машинного кода и представления его на ассемблере

Трансляторы делятся на:

- компиляторы – превращают текст программы в машинный код, который можно сохранить и затем использовать уже без компилятора (примером являются

исполняемые файлы с расширением *. exe).
- интерпретаторы – превращают часть программы в машинный код, выполняют и после этого переходят к следующей части. При этом каждый раз при выполнении программы используется интерпретатор[3].

Примером языка низкого уровня является ассемблер. Языки низкого уровня ориентированы на конкретный тип процессора и учитывают его особенности, поэтому для переноса программы на ассемблере на другую аппаратную платформу ее нужно почти полностью переписать. Определенные различия имеются и в синтаксисе программ под разные компиляторы. Правда, центральные процессоры для компьютеров фирм AMD и Intel практически совместимы и отличаются лишь некоторыми специфическими командами. А вот специализированные процессоры для других устройств, например, видеокарт, телефонов содержат существенные различия[4].

Преимущества

С помощью языков низкого уровня создаются эффективные и компактные программы, поскольку разработчик получает доступ ко всем возможностям процессора.

Недостатки

- Программист, работающий с языками низкого уровня, должен быть высокой квалификации, хорошо понимать устройство микропроцессорной системы, для которой создается программа. Так, если программа создается для компьютера, нужно знать устройство компьютера и, особенно, устройство и особенности работы его процессора.
- результирующая программа не может быть перенесена на компьютер или устройство с другим типом процессора.
- значительное время разработки больших и сложных программ.

Языки низкого уровня, как правило, используют для написания небольших системных программ, драйверов устройств, модулей стыков с нестандартным оборудованием, программирование специализированных микропроцессоров, когда важнейшими требованиями являются компактность, быстродействие и возможность прямого доступа к аппаратным ресурсам.

Ассемблер – язык низкого уровня, что широко применяется до сих пор[5].

Языки программирования высокого уровня

Можно сказать более понятными человеку, чем компьютеру. Особенности конкретных компьютерных архитектур в них не учитываются, поэтому созданные программы легко переносятся с компьютера на компьютер. В основном достаточно просто перекомпилировать программу под определенную компьютерную архитектурную и операционную систему. Разрабатывать программы на таких языках гораздо проще и ошибок допускается меньше. Значительно сокращается время разработки программы, что особенно важно при работе над большими программными проектами[6].

К языкам программирования высокого уровня относятся:

- Фортран
- Кобол
- Алгол
- Pascal
- Java
- C
- C++
- C#
- Objective C
- Smalltalk
- Delphi

Недостатком языков высокого уровня является больший размер программ по сравнению с программами на языке низкого уровня. Поэтому в основном языки высокого уровня используются для разработок программного обеспечения компьютеров и устройств, которые имеют большой объем памяти. А разные подвиды ассемблера применяются для программирования других устройств, где критичным является размер программы[7].

За 60 с лишним лет развития ЭВМ были разработаны сотни языков программирования, многие из которых используются и сейчас (например Бейсик и Фортран были впервые применены уже в конце 1950-х годов), ежегодно появляется несколько новых языков промышленного применения (не считая десятков экспериментальных).

Для того чтобы разобраться в них, языки классифицируют по важнейшим признакам:

- эволюционным – поколения языков;
- функциональным – по назначению, исполняемым функциям (описательные, логические, математические);
- уровню языка – то есть уровню обобщения в словах-операторах языка (низкого, среднего, высокого);
- области применения – то есть где применяется язык (системные, сетевые, встроенные и пр[8].

Все типы классификаций естественным образом пересекаются, гармонируют между собой, что мы увидим при рассмотрении этих классификаций, что при понимании этого позволит легко разобраться в любом новом языке – его назначении, возможностях, технике освоения.

Базовая иерархия языков программирования является системно-параллельной иерархией, то есть пакета тесно связанных иерархий: этапов программирования, поколений языков программирования и самих языков – протоколов преобразования структурной и алгоритмической информации: структурно-дескриптивного описания данных и алгоритма их обработки. Поэтому все языки делятся на два полярных типа: дескриптивные (декларативные) и алгоритмические (командные). Однако, так как в любом алгоритме существует необходимость описания данных и структур, а в любой конструкции — порядок её сборки, то реальные языки являются частично декларативными, а частично алгоритмическими, что отражается в наличии описательной и командной (рецептурной) частей любой компьютерной программы[9].

Рассмотрение пакета параллельных иерархий языков программирования целесообразно начинать с иерархии этапов программирования.

Этап 1.

Постановка задачи программирования - включает формализацию цели программирования, часто количественно-математическую, но всегда — формально-логическую, позволяющую осуществить все последующие этапы и достичь поставленной цели программирования после выполнения его этапов.

Этап 2.

Алгоритмизация – включает построение блок-схемы алгоритма, то есть последовательных шагов обработки данных и структуры самих данных для работы программы.

Этап 3.

«Кодирование» - (от ам. Традиционного слэнга «coding») - написание текста программы на базовом текстовом языке программирования, который может быть понятен транслятору — программе, преобразующей текст в бинарный код.

Этап 4.

Трансляция – перевод программы в бинарный «объектный» код, производимая транслятором без участия человека, не считая процесса отладки.

Этап 5.

Сборка исполняемого модуля программы – представляет собой автоматическую стыковку всех объектных модулей, необходимых для получения работающей программы — последовательности команд процессора компьютера, на котором выполняется задуманный алгоритм в виде двоичного кода, понятного процессору [10].

Первый этап программирования — это наиболее общий, высший иерархический уровень процесса программирования, а пятый — выполняемый автоматически компьютером — низший. Здесь к мету заметить, что отношение программистов к этим этапам сразу определяет и иерархическое положение программиста. Самоучки, любители, фрилансеры часто пропускают 2 первых этапа программирования, что часто превращает их труд в «мартышкин» - программа либо не работает, либо выполняет не то, что хотел программист. Грамотные техники-программисты (уровень колледжа) обычно начинают со второго этапа, так как первый — постановку задачи им дает руководитель проекта.

Перечисленные этапы программирования в точности соответствуют поколениям языков (generation of languages, GL) — иерархии компьютерных языков, только в обратном порядке[11].

1.2 Поколения языков программирования

Поколение 1GL.

Машинные языки, языки низкого уровня – двоичные языки процессоров, представляющие собой набор (алфавит) команд, записанных в двоичном коде (0,1),

которые данный процессор может выполнить непосредственно, если эти команды ввести в его память в виде последовательности или сразу подать в арифметическо-логическое устройство процессора. Примеры: язык процессора IBM-PC, язык ARM-процессора.

Поколение 2GL.

Ассемблеры, автокоды, системные языки, языки среднего уровня – текстовые языки, понятные человеку и однозначно переводимые (транслируемые) в языки низкого уровня, то есть машинный двоичный код. Программирование на 2GL на порядок производительнее, чем на 1GL, так как более удобны для человеческого восприятия[12]. Примеры: Макроссемблер, C, PL/1.

Поколение 3GL.

Языки высокого уровня – текстовые языки, приближенные по словарю и синтаксису к человеческому языку (обычно утрированному английскому, пиндосу), позволяющие записывать программные конструкции в форме, удобной для человеческого мышления и подобные обычному тексту — конспекту, стенограмме. Программирование на 3GL на порядок производительнее, чем на 2GL, так как более удобны для человеческого восприятия и на порядок короче ассемблерных. Примеры: бейсик, фортран, PHP и практически все сетевые языки.

Поколение 4GL.

Языки визуального программирования – языки блок-схем, позволяющие отображать алгоритмы в программных проектах, что облегчает создание и анализ алгоритмов. Программирование на 4GL на порядок производительнее, чем на 3GL. Примеры: RAD-системы, CAD-пакеты, OLAP-системы[13].

Поколение 5GL.

Интеллектуальные языки программирования - позволяют передать функцию создания алгоритмов компьютеру, а за человеком оставить лишь постановку задачи. Программирование на 5GL на порядок производительнее, чем на 4GL. Примеры: система MatCAD, экспертные системы.

Заметим, что увеличение производительности труда программиста оборачивается увеличением нагрузки на процессор и память, то есть компьютерная программа, полученная средствами каждого следующего поколения имеет на порядок большую длину исполняемого кода, а значит, расхода вычислительных ресурсов и

памяти компьютера, то есть выполняется намного медленнее на том же самом компьютере или требует намного более быстрого процессора.

Хотя поколения языков, естественно, сложились исторически, это не означает, что ранние поколения себя изжили. Они применяются в своих нишах, например, для программирования простейших устройств, имеющих минимум памяти, типа банковских карт, микроконтроллеров бытовых и промышленных устройств применим только машинный код, ибо языкам высоких уровней там «не развернуться». В системном программировании наилучшие результаты дают языки 2GL, ибо в этой сфере важна скорость выполнения и компактность кода. Для обработки текста и сетевых задач оптимальными являются языки 3GL[14].

Непосредственно связанной с иерархией поколений языков является так называемая «Стандартная модель OSI», описывающая 7 уровней иерархии протоколов (языков) сетевого обмена информацией, рассмотренная ниже.

Сетевая модель OSI (англ. open systems interconnection basic reference model — базовая эталонная модель взаимодействия открытых систем, сокр. ЭМВОС; 1978 год) — сетевая модель стека сетевых протоколов OSI/ISO (ГОСТ Р ИСО/МЭК 7498-1-99)[15].

В литературе наиболее часто принято начинать описание уровней модели OSI с 7-го уровня, называемого прикладным, на котором пользовательские приложения обращаются к сети. Модель OSI заканчивается 1-м уровнем — физическим, на котором определены стандарты, предъявляемые независимыми производителями к средам передачи данных:

- тип передающей среды (медный кабель, оптоволокно, радиоэфир и др.),
- тип модуляции сигнала,
- сигнальные уровни логических дискретных состояний (нуля и единицы).

Любой протокол модели OSI должен взаимодействовать либо с протоколами своего уровня, либо с протоколами на единицу выше и/или ниже своего уровня. Взаимодействия с протоколами своего уровня называются горизонтальными, а с уровнями на единицу выше или ниже — вертикальными. Любой протокол модели OSI может выполнять только функции своего уровня и не может выполнять функций другого уровня, что не выполняется в протоколах альтернативных моделей[16].

Каждому уровню с некоторой долей условности соответствует свой операнд — логически неделимый элемент данных, которым на отдельном уровне можно оперировать в рамках модели и используемых протоколов: на физическом уровне мельчайшая единица — бит, на канальном уровне информация объединена в кадры, на сетевом — в пакеты (датаграммы), на транспортном — в сегменты. Любой фрагмент данных, логически объединённых для передачи — кадр, пакет, датаграмма — считается сообщением. Именно сообщения в общем виде являются операндами сеансового, представительского и прикладного уровней[17].

1.3 Функциональная классификация языков программирования

Существующие языки программирования классифицируют по четырём основным функциональным группам: процедурные, объектно-ориентированные, функциональные и логические. Дадим краткие определения каждого подхода.

Процедурное программирование – такое программирование, когда программа отделена от данных и состоит из последовательности команд, обрабатывающих данные. Данные как правило хранятся в виде переменных. Весь процесс вычисления сводится к изменению их содержимого.

Выполнение программы сводится к последовательному выполнению операторов с целью преобразования исходного состояния памяти, то есть значений исходных данных, в заключительное, то есть в результаты. Таким образом, с точки зрения программиста имеются программа и память, причем первая последовательно обновляет содержимое последней.

Процедурный язык программирования предоставляет возможность программисту определять каждый шаг в процессе решения задачи. Особенность таких языков программирования состоит в том, что задачи разбиваются на шаги и решаются шаг за шагом. Используя процедурный язык, программист определяет языковые конструкции для выполнения последовательности алгоритмических шагов[18].

Декларативные языки программирования – это языки объявлений и построения структур. К ним относятся функциональные и логические языки программирования. В этих языках не производится алгоритмических действий явно, то есть алгоритм не задается прграммистом, а строится самой программой. В декларативных языках задается, производится построение какой-либо структуры

или системы, то есть декларируются (объявляются) какие-то свойства создаваемого объекта. Эти языки получили широкое применение в системах автоматизированного проектирования (САПР), в так называемых CAD-пакетах, в моделировнии, системах исккусственного интеллекта.

Функциональное программирование – это способ составления программ, в которых единственным действием является вызов функции. В функциональном программировании не используется память, как место для хранения данных, а, следовательно, не используются промежуточные переменные, операторы присваивания и циклы. Ключевым понятием в функциональных языках является выражение. Программа, написанная на функциональном языке, представляет собой последовательность описания функций и выражений. Выражение вычисляется сведением сложного к простому. Все выражения записываются в виде списков.

Первым языком стал язык Лисп (LISP, LIST Processing- обработка списков) создан в 1959г. Этот язык позволяет обрабатывать большие объемы текстовой информации. Логическое программирование- это программирование в терминах логики. В 1973 году был создан язык искусственного интеллекта Пролог (PROLOG) (Programming in Logic). Программа на языке Пролог строится из последовательности фактов и правил, затем формулируется утверждение, которое Пролог пытается доказать с помощью правил. Язык сам ищет решение с помощью методов поиска и сопоставления, которые в нем заложены. Логические программы не отличаются высоким быстродействием, так как процесс их выполнения сводится к построению прямых и обратных цепочек рассуждений разнообразными методами поиска[19].

Объектно-ориентированное программирование - в этих языках переменные и функции группируются в так называемые классы (шаблоны). Благодаря этому достигается более высокий уровень структуризации программы. Объекты, порождённые от классов вызывают методы (функции или процедуры) друг друга и меняют таким образом состояние свойств (переменных). С формально-математической стороны объектно ориентированный способ написания программ базируется на процедурной модели программирования, но с содержательной стороны ООП базируется не на функции, а на объекте, как целостной системе, имеющей стандартный автоматический межобъектный интерфейс.

Ключевые черты ООП хорошо известны:

1. Первая — инкапсуляция — это определение классов — пользовательских типов данных, объединяющих своё содержимое в единый тип и реализующих некоторые операции или методы над ним. Классы обычно являются основой модульности, инкапсуляции и абстракции данных в языках ООП.
2. Вторая ключевая черта, — наследование — способ определения нового типа, когда новый тип наследует элементы (свойства и методы) существующего, модифицируя или расширяя их. Это способствует выражению специализации и генерализации.
3. Третья черта, известная как полиморфизм, позволяет единообразно ссылаться на объекты различных классов (обычно внутри некоторой иерархии). Это делает классы ещё удобнее и облегчает расширение и поддержку программ, основанных на них[20].

Инкапсуляция, наследование и полиморфизм — фундаментальные свойства, которыми должен обладать язык, претендующий называться объектно-ориентированным (языки, не имеющие наследования и полиморфизма, но имеющие только классы, обычно называются основанными на классах). Различные ОО языки используют совершенно разные подходы. Мы можем различать ОО языки, сравнивая механизм контроля типов, способность поддерживать различные программные модели и то, какие объектные модели они поддерживают.

Сетевые языки – языки, предназначенные для организации взаимодействия удаленных компьютеров в интенсивном интерактивном режиме, а поэтому они построены на принципах интерпретации, то есть построчной, интерактивной обработки строк программного кода, описывающего некоторый сценарий (скрипт) сетевого взаимодействия компьютеров, поэтому часто они называются скриптовыми языками, хотя скриптовые языки не обязательно являются сетевыми, к примеру, пакетные командные языки различных операционных сред.

Программа или процесс, инициирующие установление связи, называются клиентским процессом, а программа, ожидающая инициации связи, называется серверным процессом. Клиентский и серверный процессы вместе образуют распределенную систему. Связь между клиентским и серверным процессами может быть или на основе соединений (как например, TCP-протокол, устанавливающий виртуальное соединение или сессию), или без соединений (на основе UDP-датаграмм)[21].

1.4 Машинно - ориентированные языки

Машинно - ориентированные языки - это языки, наборы операторов и изобразительные средства которых существенно зависят от особенностей ЭВМ (внутреннего языка, структуры памяти и т.д.). Машинно -ориентированные языки позволяют использовать все возможности и особенности Машинно - зависимых языков:

- высокое качество создаваемых программ (компактность и скорость выполнения);
- возможность использования конкретных аппаратных ресурсов;
- предсказуемость объектного кода и заказов памяти;
- для составления эффективных программ необходимо знать систему команд и особенности функционирования данной ЭВМ;
- трудоемкость процесса составления программ (особенно на машинных языках и ЯСК), плохо защищенного от появления ошибок;
- низкая скорость программирования;
- невозможность непосредственного использования программ, составленных на этих языках, на ЭВМ других типов[22].

Машинно-ориентированные языки по степени автоматического программирования подразделяются на классы.

Машинный язык

Как уже упоминалось в введении, отдельный компьютер имеет свой определенный Машинный язык (далее МЯ), ему предписывают выполнение указываемых операций над определяемыми ими операндами, поэтому МЯ является командным. Однако, некоторые семейства ЭВМ (например, ЕС ЭВМ, IBM-370 и др.) имеют единый МЯ для ЭВМ разной мощности. В команде любого из них сообщается информация о местонахождении операндов и типе выполняемой операции.

В новых моделях ЭВМ намечается тенденция к повышению внутренних языков машинно - аппаратным путем реализовывать более сложные команды, приближающиеся по своим функциональным действиям к операторам алгоритмических языков программирования[23].

Языки Символического Кодирования

Языки Символического Кодирования (далее ЯСК), так же, как и МЯ, являются командными. Однако коды операций и адреса в машинных командах, представляющие собой последовательность двоичных (во внутреннем коде) или восьмеричных (часто используемых при написании программ) цифр, в ЯСК заменены на символы (идентификаторы), форма написания которых помогает программисту легче запоминать смысловое содержание операции. Это обеспечивает существенное уменьшение числа ошибок при составлении программ.

Использование символических адресов – первый шаг к созданию ЯСК. Команды ЭВМ вместо истинных (физических) адресов содержат символические адреса. По результатам составленной программы определяется требуемое количество ячеек для хранения исходных промежуточных и результирующих значений. Назначение адресов, выполняемое отдельно от составления программы в символических адресах, может проводиться менее квалифицированным программистом или специальной программой, что в значительной степени облегчает труд программиста[24].

Автокоды

Есть также языки, включающие в себя все возможности ЯСК, посредством расширенного введения макрокоманд – они называются Автокоды.

В различных программах встречаются некоторые достаточно часто использующиеся командные последовательности, которые соответствуют определенным процедурам преобразования информации. Эффективная реализация таких процедур обеспечивается оформлением их в виде специальных макрокоманд и включением последних в язык программирования, доступный программисту. Макрокоманды переводятся в машинные команды двумя путями - расстановкой и генерированием. В постановочной системе содержатся "остовы" - серии команд, реализующих требуемую функцию, обозначенную макрокомандой. Макрокоманды обеспечивают передачу фактических параметров, которые в процессе трансляции вставляются в "остов" программы, превращая её в реальную машинную программу.

В системе с генерацией имеются специальные программы, анализирующие макрокоманду, которые определяют, какую функцию необходимо выполнить и формируют необходимую последовательность команд, реализующих данную функцию.

Обе указанных системы используют трансляторы с ЯСК и набор макрокоманд, которые также являются операторами автокода.

Развитые автокоды получили название Ассемблеры. Сервисные программы и пр., как правило, составлены на языках типа Ассемблер.

Макрос

Язык, являющийся средством для замены последовательности символов описывающих выполнение требуемых действий ЭВМ на более сжатую форму - называется Макрос (средство замены).

В основном, Макрос предназначен для того, чтобы сократить запись исходной программы. Компонент программного обеспечения, обеспечивающий функционирование макросов, называется макропроцессором. На макропроцессор поступает макроопределяющий и исходный текст. Реакция макропроцессора на вызов-выдача выходного текста.

Макрос одинаково может работать, как с программами, так и с данными[25].

1.5 Машинно - независимые языки

Машинно - независимые языки - это средство описания алгоритмов решения задач и информации, подлежащей обработке. Они удобны в использовании для широкого круга пользователей и не требуют от них знания особенностей организации функционирования ЭВМ и ВС[26].

Подобные языки получили название высокоуровневых языков программирования. Программы, составляемые на таких языках, представляют собой последовательности операторов, структурированные согласно правилам рассматривания языка(задачи, сегменты, блоки и т.д.). Операторы языка описывают действия, которые должна выполнять система после трансляции программы на МЯ.

Т.о., командные последовательности (процедуры, подпрограммы), часто используемые в машинных программах, представлены в высокоуровневых языках отдельными операторами. Программист получил возможность не расписывать в деталях вычислительный процесс на уровне машинных команд, а сосредоточиться на основных особенностях алгоритма.

Проблемно - ориентированные языки

С расширением областей применения вычислительной техники возникла необходимость формализовать представление постановки и решение новых классов задач. Необходимо было создать такие языки программирования, которые, используя в данной области обозначения и терминологию, позволили бы описывать требуемые алгоритмы решения для поставленных задач, ими стали проблемно - ориентированные языки. Эти языки, ориентированные на решение определенных проблем, должны обеспечить программиста средствами, позволяющими коротко и четко формулировать задачу и получать результаты в требуемой форме[27].

Проблемных языков очень много, например:

- Фортран, Алгол – языки, созданные для решения математических задач;
- Simula, Слэнг – для моделирования;
- Лисп, Снобол – для работы со списочными структурами.
- Об этих языках рассказано дальше.

Универсальные языки

Универсальные языки были созданы для широкого круга задач: коммерческих, научных, моделирования и т.д. Первый универсальный язык был разработан фирмой IBM, ставший в последовательности языков Пл/1. Второй по мощности универсальный язык называется Алгол-68. Он позволяет работать с символами, разрядами, числами с фиксированной и плавающей запятой. Пл/1 имеет развитую систему операторов для управления форматами, для работы с полями переменной длины, с данными организованными в сложные структуры, и для эффективного использования каналов связи. Язык учитывает включенные во многие машины возможности прерывания и имеет соответствующие операторы. Предусмотрена возможность параллельного выполнение участков программ.

Программы в Пл/1 компилируются с помощью автоматических процедур. Язык использует многие свойства Фортрана, Алгола, Кобола. Однако он допускает не только динамическое, но и управляемое и статистическое распределения памяти [28].

Диалоговые языки

Появление новых технических возможностей поставило задачу перед системными программистами – создать программные средства, обеспечивающие оперативное взаимодействие человека с ЭВМ их назвали диалоговыми языками.

Эти работы велись в двух направлениях. Создавались специальные управляющие языки для обеспечения оперативного воздействия на прохождение задач, которые составлялись на любых раннее неразработанных (не диалоговых) языках. Разрабатывались также языки, которые кроме целей управления обеспечивали бы описание алгоритмов решения задач.

Необходимость обеспечения оперативного взаимодействия с пользователем потребовала сохранения в памяти ЭВМ копии исходной программы даже после получения объектной программы в машинных кодах. При внесении изменений в программу с использованием диалогового языка система программирования с помощью специальных таблиц устанавливает взаимосвязь структур исходной и объектной программ. Это позволяет осуществить требуемые редакционные изменения в объектной программе.

Одним из примеров диалоговых языков является Бэйсик.

Бэйсик использует обозначения подобные обычным математическим выражениям. Многие операторы являются упрощенными вариантами операторов языка Фортран. Поэтому этот язык позволяет решать достаточно широкий круг задач[29].

Непроцедурные языки

Непроцедурные языки составляют группу языков, описывающих организацию данных, обрабатываемых по фиксированным алгоритмам (табличные языки и генераторы отчетов), и языков связи с операционными системами.

Позволяя четко описывать как задачу, так и необходимые для её решения действия, таблицы решений дают возможность в наглядной форме определить, какие условия должны быть выполнены прежде чем переходить к какому-либо действию. Одна таблица решений, описывающая некоторую ситуацию, содержит все возможные блок-схемы реализаций алгоритмов решения.

Табличные методы легко осваиваются специалистами любых профессий.

Программы, составленные на табличном языке, удобно описывают сложные ситуации, возникающие при системном анализе.

Непроцедурные языки – современное направление системного программирования, позволяющее сконцентрировать внимание разработчика на описании целей и правил, а не на последовательности действий по их реализации (т.е. описывается «что делать» вместо «как делать»). Использование непроцедурного языка

обеспечивает максимально возможную простоту и понятность программ для разработчика-технолога, перенося сложности процедурной реализации на системный уровень, что сокращает трудоемкость и сроки разработки, увеличивает надежность ПО.

1.6 Становление непроцедурного программирования

Как известно, человеческое мышление построено прежде всего на ассоциациях и связях. Естественное мышление непроцедурно - алгоритмы типа «делай раз, делай два» составляются с трудом лучшими специалистами, а всеми остальными выучиваются наизусть. Вне производства люди по-возможности избегают жестких алгоритмов (показательно, что даже в классически-процедурном кулинарном процессе хозяйки не любят использовать «строгие» измерения и активно варьируют рецептуру, сохраняя смысловые связи между компонентами).

Однако, современные вычислительные машины требуют процедурного описания своего функционирования. Поэтому, уже в течении последних 50 лет все большее количество людей занимаются ручной компиляцией своего непроцедурного мышления в процедурные описания. Принципиальное несовершенство глобальной процедурности было осознано сравнительно недавно, а системы непроцедурного программирования, позволяющие заменить ручную компиляцию знаний на автоматическую, только начинают заявлять о себе. В настоящий момент такие системы активно используются в элитных отраслях типа космических исследований, параллельных вычислений в задачах ядерной физики и т.п. Эпоха широкого внедрения в непроцедурного программирования в промышленности еще не наступила, хотя в лабораториях крупных фирм ведутся соответствующие разработки[30].

Одной из принципиальных особенностей становления непроцедурного программирования является преимущественное развитие неуниверсальных языков, ориентированных на конкретную предметную область (в нашем случае - на АСУТП). Универсальность является неизбежной жертвой дружественности языка для технолога. Как говорят специалисты, «универсальный язык дружественен для системного программиста». Впрочем, традиционные процедурные языки технологического программирования также неуниверсальны.

Совокупность объектно-компонентных технологий дала качественно новый базис для построения систем непроцедурного программирования, что существенно упростило и удешевило их создание, открывая возможность использования в базовых отраслях промышленности.

Заканчивая исторический экскурс хотелось бы особо отметить, что непроцедурное программирование – одна из немногих областей, в которых отечественная наукоемкое производство находится на передовых рубежах. Несколько коллективов РАН, ведущих соответствующие разработки высоко котируются в мире. К сожалению, Российское правительство практически не финансирует развитие этого стратегического направления отечественной научной технологии, а существенным источником финансирования работ в течении длительного времени были заказы от НАСА и министерства обороны США[31].

2. Критерии выбора среды и языка разработки программ

2.1 Критерии выбора среды разработки программ

Среды программирования (или как их еще называют, среды разработки) – это программы, в которых программисты пишут свои программы. Иными словами, среда программирования служит для разработки (написания) программ и обычно ориентируется на конкретный язык или несколько языков программирования (в этом случае языки, обычно, принадлежат одной языковой группе, например, Си-подобные). Компоненты среды программирования:

1. Редактор – это средство для создания и изменения исходных файлов, которые содержат написанную на языке программирования программу.

Условно редакторы делятся на два типа.

Первый тип работает с последовательностью символов в текстовых файлах и обеспечивают расширенную функциональность – подсветку синтаксиса, сортировку строк, конвертацию кодировок, показ кодов символов и т.п. Часто такие редакторы называют редакторами кода, поскольку основное их назначение – это написание исходных кодов компьютерных программ. Примеры таких редакторов: Emacs (один

из самых мощных по возможностям многоцелевой, свободный редактор); jEdit (свободный редактор на Java); Kate (мощный расширяемый свободный текстовый редактор с подсветкой синтаксиса для массы языков программирования и разметки); Notepad (входит в состав Microsoft Windows); Vim (один из самых мощных по возможностям редактор для программистов); EditPlus (текстовый редактор для Windows, предназначенный для программирования и веб-разработки) и др.

Второй тип редакторов имеет расширенные функции форматирования текста, внедрения в него графики и формул, таблиц и объектов. Такие редакторы часто называют текстовыми процессорами и предназначены они для создания текстовых документов. К таким текстовым процессорам можно отнести Microsoft Word, WordPad и др.

1. Компилятор – транслирует символы из исходного файла в объектный модуль, который содержит команды в машинном коде для конкретного компьютера.
2. Компоновщик или редактор связей – собирает объектные файлы отдельных компонентов программы и разрешает внешние ссылки от одного компонента к другому, формируя исполняемый файл.
3. Загрузчик – копирует исполняемый файл с диска в память и инициализирует компьютер перед выполнением программы.
4. Отладчик – это средство, которое дает возможность программисту управлять выполнением программы на уровне отдельных операторов для диагностики ошибок. Позволяет выполнять пошаговую трассировку (пошаговое выполнение программы с остановками на каждой команде или строке), отслеживать, устанавливать или изменять значения переменных в процессе выполнения программы, устанавливать и удалять контрольные точки или условия остановки и т.д.
5. Средства тестирования – автоматизирует процесс тестирования программ, создавая и выполняя тесты и анализируя результаты тестирования.
6. Интерпретатор – выполняет исходный код программы в отличие от компилятора, переводящего исходный файл в объектный[32].

2.2 Критерии выбора языка разработки программ

Язык программирования — формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и

действия, которые выполнит исполнитель (обычно — ЭВМ) под её управлением.

Со времени создания первых программируемых машин человечество придумало более восьми тысяч языков программирования (включая эзотерические, визуальные и игрушечные). Каждый год их число увеличивается. Некоторыми языками умеет пользоваться только небольшое число их собственных разработчиков, другие становятся известны миллионам людей. Профессиональные программисты могут владеть десятком и более разных языков программирования [33].

Язык программирования предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной вычислительный процесс, организовать управление различными объектами, и т. п. Язык программирования отличается от естественных языков тем, что предназначен для управления ЭВМ, в то время как естественные языки используются, прежде всего, для общения людей между собой. Большинство языков программирования использует специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений.

Как правило, язык программирования определяется не только через спецификации стандарта языка, формально определяющие его синтаксис и семантику, но и через воплощения (реализации) стандарта — программные средства, обеспечивающих трансляцию или интерпретацию программ на этом языке; такие программные средства различаются по производителю, марке и варианту (версии), времени выпуска, полноте воплощения стандарта, дополнительным возможностям; могут иметь определённые ошибки или особенности воплощения, влияющие на практику использования языка или даже на его стандарт.

На сегодняшний день существует широкий выбор различных средств реализации программного кода. Используемый в проекте язык программирования определяет скорость разработки и реализации, простоту сопровождения, возможность переноса создаваемого в рамках этого проекта программного обеспечения.

По мере возрастания роли программного обеспечения и требований к нему программистами были созданы различные языки программирования предоставляющие повышение эффективности, надежности, мобильности и упрощение написанного с их помощью программного обеспечения. Появление множества различных языков программирования требуют тщательного выбора единственного правильного варианта для решения конкретной задачи. На этапе

реализации программного кода выполняется кодирование отдельных элементов программы согласно техническому заданию.

В связи с этим, перед разработчиками стоит вопрос - какой язык реализации выбрать? Для начала происходит отбор критериев важных для конкретной сферы деятельности, после чего происходит непосредственно выбор подходящего варианта[34].

Когда необходимо создать большую программную систему или составить программы для решения какой-либо частной задачи, в том числе при создании компьютерных игр, встает вопрос, какой выбрать для этой цели наиболее подходящий язык программирования. В большинстве случаев такой выбор делается на основании очень простых «земных» факторов: наличии того или иного транслятора и умения делать софт на данном языке. Но если в распоряжении пользователя несколько языков программирования и нужно создать, то необходимо учитывать следующие обстоятельства:

— назначение разрабатываемого софта, то есть необходимость в ней будет временная или она будет использоваться постоянно, будет ли она в дальнейшем передаваться другим организациям, будут ли создаваться ее новые версии;

— необходимая скорость работы софта, соотношение и работа его вычислительных и диалоговых компонентов;

— предполагаемый размер программы, то есть создавать ее как единое целое или она будет в виде отдельных взаимодействующих модулей, нужно ли минимизировать размер памяти, которую занимает программа во время работы;

— возможность сопряжения разрабатываемого софта с другими приложениями (пакетами или программами), включая приложения, составленные на иных языках программирования;

— основные типы данных, которыми придется оперировать, возможность поддержки работы с различными типами структур (строками, действительными числами, списками и др.);

— характер и уровень использования периферийных средств (монитора, клавиатуры и др.), необходимость в специальном программировании некоторых функций, чтобы работать с периферийными устройствами;

— целесообразность и возможность применения имеющихся стандартных библиотек подпрограмм, процедур, функций[35].

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была достигнута поставленная цель: изучена классификация языков программирования, произведен анализ критериев выбора среды и языка программирования.

Для достижения данной цели необходимо было выполнить следующие задачи:

1. рассмотреть классификацию языков программирования;
2. изучить критерии выбора среды разработки;
3. проанализировать критерии выбора языка разработки.

Разработка множества языков программирования высшего уровня, а также их непрерывная модернизация и усовершенствование, предоставило человечеству возможность не только реализовывать выгодное общение с машиной, но и применять технические и информационные средства для реализации самых сложных расчетов в области самолетостроения, ракетостроения, медицины и даже экономики.

Вне зависимости от того, что настоящий уровень развития языков программирования достиг непостижимых высот, направление их развития и совершенствования дают нам возможность предполагать, что в скором времени будут созданы такие языки программирования, о которых сейчас человечество не может и мечтать.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. C++,Turbo Pasckal,QBasik: Эволюция языков программирования http://langprog.far.ru/historylangprog.html. 2010
2. Будников А.И. Сравнительный анализ производительности реализаций инструментария Qt для языков C++ и Python // Новосибирск: ООО "Центр развития научного сотрудничества". – 2014 – 129-132 с.
3. Граничин О.Н., Кияев В.И. Информационные технологии и системы в современном менеджменте. – СПб.: Издательство ВВМ. – 2014. – 897 с.

4. Курносов А.П., Кулев С.А., Улезько А.В. и др.; Информатика Под ред. А.П. Курносова.-М.: КолосС, 2005.-272 с
5. Макарова Н.В. Информатика /под ред. Проф. Н.В. Макаровой. — М.: Финансы и статистика, 1997. — 768 с.: ил.
6. Маккинли У. Python и анализ данных // пер. с англ. Слинкин А. А. - М.: ДМК Пресс. - 2015. – 482 с.
7. Малышев Р.А. Локальные вычислительные сети: Учебное пособие/ РГАТА. – Рыбинск, 2005. – 83 с.
8. Марманис Х., Бабенко Д. Алгоритмы интеллектуального Интернета. Передовые методики сбора, анализа и обработки данных // пер. с англ. – СПБ.:Символ-Плюс.– 2011.– 480 с.
9. Островский В.А. Информатика: учеб. для вузов. М.: Высшая школа, 2000. —511 с.:
10. Саати Т. Принятие решений методом анализа иерархий // М.:Радио-Связь. – 1994 – 278 с.
11. Семакин И.А., Информатика: Базовый курс /Семакин И.А., Залогова Л., Русаков С., Шестакова Л. – Москва: БИНОМ.,2005. – 105с.
12. Симонович С.В.Информатика. Базовый курс/Симонович С.В. и др. — СПб.: издательство "Питер", 2000. — 640 с.: ил.
13. Смирнов В.А. Новые компетенции социолога в эпоху «больших данных» // Мониторинг общественного мнения: экономически и социальные перемены.: Всероссийский центр изучения общественного мнения (Москва). - №2. – 2015 – 44-54 с.
14. Трофимов В.В. Информационные технологии в экономике и управлении. М.: Юрайт, 2014. – 482 с.
15. Федеральный закон от 13.07.2015 «О деятельности кредитных рейтинговых агентств в Российской, внесении изменения в статью 76.1 федерального закона «О центральном банке Российской Федерации (Банке России)»».

1. Семакин И.А., Информатика: Базовый курс /Семакин И.А., Залогова Л., Русаков С., Шестакова Л. – Москва: БИНОМ.,2005. – 64с. ↑

2. Трофимов В.В. Информационные технологии в экономике и управлении. М.: Юрайт, 2014. – 129 с. ↑

3. Островский В.А. Информатика: учеб. для вузов. М.: Высшая школа, 2000. —301 с ↑

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

5. Курносов А.П., Кулев С.А., Улезько А.В. и др.; Информатика Под ред. А.П. Курносова.-М.: КолосС, 2005.-129 с ↑

6. Курносов А.П., Кулев С.А., Улезько А.В. и др.; Информатика Под ред. А.П. Курносова.-М.: КолосС, 2005.-76 с ↑

7. Курносов А.П., Кулев С.А., Улезько А.В. и др.; Информатика Под ред. А.П. Курносова.-М.: КолосС, 2005.-81 с ↑

8. Макарова Н.В. Информатика /под ред. Проф. Н.В. Макаровой. — М.: Финансы и статистика, 1997. — 567 с ↑

9. Марманис Х., Бабенко Д. Алгоритмы интеллектуального Интернета. Передовые методики сбора, анализа и обработки данных // пер. с англ. – СПБ.:Символ-Плюс.– 2011.– 178 с ↑

10. Саати Т. Принятие решений методом анализа иерархий // М.:Радио-Связь. – 1994 – 207 ↑

11. Саати Т. Принятие решений методом анализа иерархий // М.:Радио-Связь. – 1994 – 213 ↑

12. Курносов А.П., Кулев С.А., Улезько А.В. и др.; Информатика Под ред. А.П. Курносова.-М.: КолосС, 2005.-188 ↑

13. Курносов А.П., Кулев С.А., Улезько А.В. и др.; Информатика Под ред. А.П. Курносова.-М.: КолосС, 2005.-193 ↑

14. Курносов А.П., Кулев С.А., Улезько А.В. и др.; Информатика Под ред. А.П. Курносова.-М.: КолосС, 2005.-197 ↑

15. Граничин О.Н., Кияев В.И. Информационные технологии и системы в современном менеджменте. – СПб.: Издательство ВВМ. – 2014. – 607 ↑

16. Маккинли У. Python и анализ данных // пер. с англ. Слинкин А. А. - М.: ДМК Пресс. - 2015. – 354 ↑

17. Семакин И.А., Информатика: Базовый курс /Семакин И.А., Залогова Л., Русаков С., Шестакова Л. – Москва: БИНОМ.,2005. – 97с ↑

18. Трофимов В.В. Информационные технологии в экономике и управлении. М.: Юрайт, 2014. – 354 с. ↑

19. Симонович С.В.Информатика. Базовый курс/Симонович С.В. и др. — СПб.: издательство "Питер", 2000. — 512 с ↑

20. Смирнов В.А. Новые компетенции социолога в эпоху «больших данных» // Мониторинг общественного мнения: экономически и социальные перемены.: Всероссийский центр изучения общественного мнения (Москва). - №2. – 2015 – 44-54 с ↑

21. Трофимов В.В. Информационные технологии в экономике и управлении. М.: Юрайт, 2014. – 123 с. ↑

22. Трофимов В.В. Информационные технологии в экономике и управлении. М.: Юрайт, 2014. – 111 с. ↑

23. Будников А.И. Сравнительный анализ производительности реализаций инструментария Qt для языков C++ и Python // Новосибирск: ООО "Центр развития научного сотрудничества". – 2014 – 129-132 ↑

24. Будников А.И. Сравнительный анализ производительности реализаций инструментария Qt для языков C++ и Python // Новосибирск: ООО "Центр развития научного сотрудничества". – 2014 – 117 ↑

25. Малышев Р.А. Локальные вычислительные сети: Учебное пособие/ РГАТА. – Рыбинск, 2005. – 81 с ↑

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

27. Островский В.А. Информатика: учеб. для вузов. М.: Высшая школа, 2000. —453 с ↑

28. Островский В.А. Информатика: учеб. для вузов. М.: Высшая школа, 2000. —500 с ↑

29. Симонович С.В.Информатика. Базовый курс/Симонович С.В. и др. — СПб.: издательство "Питер", 2000. — 287 с ↑

30. Симонович С.В.Информатика. Базовый курс/Симонович С.В. и др. — СПб.: издательство "Питер", 2000. — 600 с ↑

31. Симонович С.В.Информатика. Базовый курс/Симонович С.В. и др. — СПб.: издательство "Питер", 2000. — 610 с ↑

32. Симонович С.В.Информатика. Базовый курс/Симонович С.В. и др. — СПб.: издательство "Питер", 2000. —123 с ↑

33. Трофимов В.В. Информационные технологии в экономике и управлении. М.: Юрайт, 2014. – 450 с. ↑

34. Семакин И.А., Информатика: Базовый курс /Семакин И.А., Залогова Л., Русаков С., Шестакова Л. – Москва: БИНОМ.,2005. – 78с. ↑

35. Семакин И.А., Информатика: Базовый курс /Семакин И.А., Залогова Л., Русаков С., Шестакова Л. – Москва: БИНОМ.,2005. – 90с. ↑