**Содержание**

Введение ……………………………………………………………………....2

1 Язык и система программирования – понятие, сущность ……………….4

2 Классификация языков программирования……………………………….6

 2.1 Машинно – ориентированные языки ………………………………....6

 2.1.1 Машинные языки ………………………………………………...6

 2.1.2 Языки символического кодирования …………………………...7

 2.1.3 Автокоды …………………………………………………………8

 2.1.4 Макрос …………………………………………………………….9

 2.2 Машинно – независимые языки ………………………………………..9

 2.2.1 Машинно – независимые языки …………………………………10

 2.2.2 Универсальные языки ……………………………………………10

 2.2.3 Диалоговые языки ………………………………………………...11

 2.2.4 Непроцедурные языки ……………………………………………12

3 Современные языки и системы программирования ………………………13

 3.1 Basic ………………………………………………………………………13

 3.2 Pascal ……………………………………………………………………...14

 3.3 Delphi ……………………………………………………………………..15

 3.4 Fortran …………………………………………………………………….17

 3.5 С и С++ …………………………………………………………………...18

 3.6 Java ………………………………………………………………………..20

Заключение ……………………………………………………………………..22

Список использованных источников ...............................................................23

**Введение**

Прогресс компьютерных технологий определил процесс появления новых разнообразных знаковых систем для записи алгоритмов – языков программирования. Смысл появления такого языка – оснащенный набор вычислительных формул дополнительной информации, превращает данный набор в алгоритм. Язык программирования служит двум связанным между собой целям: он дает программисту аппарат для задания действий, которые должны быть выполнены, и формирует концепции, которыми пользуется программист, размышляя о том, что делать. Первой цели идеально отвечает язык, который настолько "близок к машине", что всеми основными машинными аспектами можно легко и просто оперировать достаточно очевидным для программиста образом. Второй цели идеально отвечает язык, который настолько "близок к решаемой задаче", чтобы концепции ее решения можно было выражать прямо и коротко. Связь между языком, на котором мы думаем/программируем, и задачами и решениями, которые мы можем представлять в своем воображении, очень близка. По этой причине ограничивать свойства языка только целями исключения ошибок программиста в лучшем случае опасно. Как и в случае с естественными языками, есть огромная польза быть, по крайней мере, двуязычным. Язык предоставляет программисту набор концептуальных инструментов, если они не отвечают задаче, то их просто игнорируют. Например, серьезные ограничения концепции указателя заставляют программиста применять вектора и целую арифметику, чтобы реализовать структуры, указатели и т.п. Хорошее проектирование и отсутствие ошибок не может гарантироваться чисто за счет языковых средств. Может показаться удивительным, но конкретный компьютер способен работать с программами, написанными на его родном машинном языке. Существует почти столько же разных машинных языков, сколько и компьютеров, но все они суть разновидности одной идей простые операции производятся со скоростью молнии на двоичных числах. Персональные компьютеры IBM используют машинный язык микропроцессоров семейства 8086, т.к. их аппаратная часть основывается именно на данных микропроцессорах. Можно писать программы непосредственно на машинном языке, хотя это и сложно. На заре компьютеризации(в начале 1950-х г.г.), машинный язык был единственным языком, большего человек к тому времени не придумал. Для спасения программистов от сурового машинного языка программирования, были созданы языки высокого уровня (т.е. немашинные языки), которые стали своеобразным связующим мостом между человеком и машинным языком компьютера. Языки высокого уровня работают через трансляционные программы, которые вводят "исходный код" (гибрид английских слов и математических выражений, который считывает машина), и в конечном итоге заставляет компьютер выполнять соответствующие команды, которые даются на машинном языке. Существует два основных вида трансляторов: интерпретаторы, которые сканируют и проверяют исходный код в один шаг, и компиляторы, которые сканируют исходный код для производства текста программы на машинном языке, которая затем выполняется отдельно.

 **1 Язык и система программирования – понятие, сущность**

 В настоящее время наблюдается стремительное развитие научной дисциплины, называемой программированием. При этом появляются не просто новые языки, появляются новые идеи, увеличивающие мощность и эффективность языков. Можно, не вдаваясь в подробности любого из существующих или только разрабатываемых языков, отметить следующую тенденцию: развитие языков идет в сторону повышения выразительности исходного текста программы. Это способствует сокращению размера программы и повышению ее надежности .

 Для повышения выразительности языка необходимо, чтобы язык содержал средства для выражения абстрактных понятий. Это помогает сделать большие программы более простыми для понимания. Поэтому поддержка абстракций является обязательным условием для любого современного языка программирования. При этом базис языка (множество предоставляемых языком возможностей, смысловых конструкций) должен иметь минимальную мощность.

 К наиболее общим понятиям, которыми оперирует программист при использовании конкретного языка программирования, относятся понятия программы и виртуальной машины. Программа должна удовлетворять требованиям (спецификациям) конкретного языка программирования и служит контейнером для хранения последовательности действий и множества данных. Виртуальная машина выступает в роли интерпретатора основных понятий, используемых в языке программирования и является средой существования программы. Все остальные абстракции, рассматриваемые в статье, группируются вокруг этих базовых абстракций.

 В ряде случаев можно рассматривать процесс программирования как процесс моделирования. При этом создается программа-модель, способная реализовывать поведение оригинала, описываемого в постановке задачи. Поэтому в дальнейшем заменителем для понятия программа будет выступать понятие модель, а для понятия виртуальная машина - понятие моделирующая среда.

 **2 Классификация языков программирования**

 **2.1 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.

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

 **2.1.1 Машинный язык**

Как я уже упоминал, 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 моднлях ЭВМ намечается тенденция к повышению внутренних языков машинно – аппаратным путем реализовывать более сложные команды, приближающиеся по своим функциональным действиям к операторам алгоритмических языков программирования.

 **2.1.2 Языки Символического Кодирования**

Продолжим рассказ о командных языках, 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.

 **2.1.3Автокоды**

 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. Более полная информация об языке Ассемблера см. ниже.

 **2.1.4 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.

 **2.2 Машинно – независимые языки**

Машинно – независимые языки – это средство описания алгоритмов решения задач и информации, подлежащей обработке [8,10,12]. Они удобны в использовании для широкого круга пользователей и не требуют от них знания особенностей организации функционирования ЭВМ и ВС.

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.

 **2.2.1 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, языки ориентированные на решение определенных проблем, должны обеспечить программиста средствами, позволяющими коротко и четко формулировать задачу и получать результаты в требуемой форме.

 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 я расскажу дальше.

 **2.2.2 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.

 **2.2.3 Диалоговые языки**

Появление новых технических возможностей поставило задачу перед системными программистами – создать программные средства, обеспечивающие оперативное взаимодействие человека с ЭВМ их назвали диалоговыми языками [10,11].

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 Бейсик.

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

 **2.2.4** **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.

 **3 Современные языки и системы программирования**

 **3.1 Basic**

Как знаменитые гамбургеры, бейсбол и баскетбол, Бейсик - это продукт Новой Англии [4]. Как я говорил, созданный в 1964г., как язык обучения программированию. Бейсик является общепринятым акронимом от"Beginner's All-purpose Symbolic Insruction Code" (BASIC) - Многоцелевой Символический Обучающий Код для Начинающих".

Вскоре как обучаемые, так и авторы программ обнаружили, что Бейсик может делать практически все то, что делает скучный неуклюжий Фортран. А так как Бейсику было легко обучиться и легко с ним работать, программы на нем писались обычно быстрее, чем на Фортране. Бейсик был также доступен на персональных компьютерах, обычно он встроен в ПЗУ. Так Бейсик завоевал популярность. Интересно, что спустя 20 лет после изобретения Бейсика, он и сегодня самый простой для освоения из десятков языков общецелевого программирования, имеющихся в распоряжении любителей программирования. Более того, он прекрасно справляется с работой.

Несмотря на высказывания снобов - сторонников языков Си и Паскаля, Бейсик считается деловым языком, снабженным мощными средствами решения специфических задач, которые обычно большинство пользователей решают при помощи небольших компьютеров, а именно: работая с файлами и выводя текстовое и графическое изображение на экране дисплея.

Несмотря на отдельные недостатки Бейсика, никто не будет отрицать, что Кемени и Куртс достигли основной цели: сделать программирование доступнее для большего числа людей.

Исторически Бейсик обычно реализовался как интерпретатор (знакомым изомером является сам интерпретаторный Бейсик). Причинами перехода от любительского уровня к профессиональному являются многочисленные расширения классической версии языка: возможность отключения нумерации строк, многостроковые структурированные программные конструкции, структуры типа "запись", поименованные подпрограммы с параметрами и локальные переменные.

Более того, с появлением транслятора QuickBasic фирмы Microsoft разработчики получили возможность строить на Бейсике приложения из раздельно откомпилированных модулей, некоторые из которых могут быть написаны на других языках. Теперь, как и в случае других ведущих языков программирования, разработчик имеет выбор из нескольких промышленных библиотек подпрограмм, которые содержат готовые решения для распространенных задач программирования.

 **3.2 Pascal**

 Язык Паскаль был создан как учебный язык программирования в 1968 –1971г. Никлаусом Виртом [8,10,11]. В настоящее время этот язык имеет более широкую сферу применения, чем предусматривалось при его создании. Целью работы Вирта было создание языка, который:

- Строился бы на небольшом количестве базовых понятий;

- Имел бы простой синтаксис;

Допускал бы перевод программ в машинный код простым компилятором;

Все эти качества сделали язык очень популярным и удобным для применения в школе.

 Паскаль – язык профессионального программирования, который назван в честь французского математика и философа Блеза Паскаля (1623–1662) и разработан в 1968–1971 гг. Николаусом Виртом, для обучения студентов методам разработки программ, таким как "программирование сверху вниз", "структурное программирование" и т. д. Вирту не понравился не один из существующих на тот момент языков, и в 1968 году он приступил к разработке своего собственного. Первая версия языка была создана для компьютера CDC 6000. Благодаря своей четкости, логичности и другим особенностям Паскаль надолго занял свою нишу, являясь прекрасным языком для обучения программированию. Паскаль использовался и для разработки серьезных программ- приложений. Шутили, что Вирт разработал игрушку, но многие отнеслись к ней слишком серьезно

 Впоследствии появились различные версии языка и его расширения. Наиболее известным расширением стал пакет Турбо Паскаль фирмы Borland, появившийся в 1983 году и сразу ставший событием в мире компьютерных технологий.

 Турбо Паскаль – это система программирования, созданная для повышения качества и скорости разработки программ (80-е гг.). Слово Турбо в названии системы программирования – это отражение торговой марки фирмы-разработчика Borland International (США).

 Систему программирования Турбо Паскаль называют интегрированной (integration – объединение отдельных элементов в единое целое) средой программирования, т.к. она включает в себя редактор, компилятор, отладчик, имеет сервисные возможности.

 Первое упоминание о нем содержалось в рекламе опубликованной в журнале Byte, а сам пакет предназначен для операционной системы CP/M. В начале 1984 года он был перенесен в среду MS-DOS и приобрел огромную популярность. С тех пор появилось несколько версий Турбо Паскаля, последняя- седьмая.

 **3.3 Delphi**

 Появление Delphi не могло пройти незамеченным среди многочисленных пользователей компьютера [7,11]. Оценки экспертов, изучающих возможности этого нового продукта фирмы Borland, обычно окрашены в восторженные тона. Основное достоинство Delphi состоит в том, что здесь реализованы идеи визуального программирования. Среда визуального программирования превращает процесс создания программы в приятное и легко понимаемое конструирование приложения из большого набора графических и структурных примитивов.

 Система Delphi позволяет решать множество задач, в частности:

* Создавать законченные приложения для Windows самой различной направленности: от чисто вычислительных и логических, до графических и мультимедиа.
* Быстро создавать (даже начинающим программистам) профессионально выглядящий оконный интерфейс для любых приложений.
* Создавать мощные системы работы с локальными и удаленными базами данных.
* Создавать справочные системы (файлы .hlp) для своих приложений и мн. др.

 Delphi – чрезвычайно быстро развивающаяся система. Первая версия – Delphi 1.0 была выпущена в феврале 1995 г. А затем новые версии выпускались ежегодно.

 Большинство версий Delphi выпускается в нескольких вариантах: Standart – стандартном, Professional – профессиональном, Client/Server – клиент/сервер, Enterprise – разработка баз данных предметных областей. Различаются варианты в основном разным уровнем доступа к системам управления базами данных. Последние варианты - Client/Server и Enterprise, в этом отношении наиболее мощные.

Delphi - это комбинация нескольких важнейших технологий:

* Высокопроизводительный компилятор в машинный код.
* Объектно-ориентированная модель компонент.
* Визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов.
* Масштабируемые средства для построения баз данных.

 **3.4 Fortran**

Одним из первых и наиболее удачных компиляторов стал язык Фортран, разработанный фирмой IBM. Профессор Дж. Букс и группа американских специалистов в области программирования в 1954 году опубликовало первое сообщение о языке. Дословно, название языка FORmulae TRANslation –преобразование формул.

Среди причин долголетия Фортрана (а он один из самых распространенных языков в мире), можно отметить простую структуру, как самого Фортрана, так и предназначенных для него трансляторов [8,10,11]. Программа на Фортране записывается в последовательности предложений или операторов (описание некоего преобразования информации), и оформляется по определенным стандартам. Эти стандарты накладывают ограничения, в частности, на форму записи и расположения частей оператора в строке бланка для записи операторов. Программа, записанная на Фортране, представляет собой один или несколько сегментов (подпрограмм) из операторов. Сегмент, управляющий работой всей программы в целом, называется основной программой.

Фортран был задуман для использования в сфере научных и инженерно-технических вычислений. Однако на этом языке легко описываются задачи с разветвленной логикой (моделирование производственных процессов, решение игровых ситуаций и т.д.), некоторые экономические задачи и особенно задачи редактирования (составление таблиц, сводок, ведомостей и т.д.).

Модификация языка Фортран, появившиеся в 1958 году, получила название Фортран II и содержала понятие подпрограммы и общих переменных для обеспечения связи между сегментами.

К 1962 году относится появление языка, известного под именем Фортран IV и ставшего наиболее употребительным в настоящее время. К этому же времени относится и начало деятельности комиссии при Американской Ассоциации Стандартов (ASA), которая выработала к 1966 году два стандарта – языки Фортран и базисный (основной) Фортран (Basic FORTRAN). Эти языки приблизительно соответствуют модификациям IV и II, однако базисный Фортран является подмножеством Фортрана, в то время, как Фортран II таковым для Фортрана IV не является. Язык Фортран до сих пор продолжает развиваться и совершенствоваться, оказывая влияние на создание и развитие других языков. Например, Фортран заложен в основу Basic – диалогового языка, очень популярного для решения небольших задач, превосходного языка для обучения навыкам использования алгоритмических языков в практике программирования. Разработан этот язык – Beginner’s All –purpose Symbolic Instruction Code – группой сотрудников Вычислительного центра Дармутского колледжа, штат Нью-Хемпшир созданный в 19…. . Но это уже следующий язык.

 **3.5 С и С++**

 Язык "C" является универсальным языком программирования. Он тесно связан с операционной системой "UNIX" , так как был развит на этой системе и так как "UNIX" и ее программное обеспечение написано на "C"[5,7,10]. Сам язык , однако, не связан с какой-либо одной операционной системой или машиной; и хотя его называют языком системного программирования, так как он удобен для написания операционных систем, он с равным успехом использовался при написании больших вычислительных программ, программ для обработки текстов и баз данных.

 Язык "C" - это язык относительно "низкогоуровня". В такой характеристике нет ничего оскорбительного; это просто означает, что "C" имеет дело с объектами того же вида, что и большинство ЭВМ, а именно, с символами, числами и адресами. Они могут объединяться и пересылаться посредством обычных арифметических и логических операций, осуществляемых реальными ЭВМ.

 В языке "C" отсутствуют операции, имеющие дело непосредственно с составными объектами, такими как строки символов, множества, списки или с массивами, рассматриваемыми как целое. Здесь, например, нет никакого аналога операциям PL/1, оперирующим с целыми массивами и строками. Язык не предос тавляет никаких других возможностей распределения памяти, кроме статического определения и механизма стеков, обеспечиваемого локальными переменных функций; здесь нет ни "куч"(HEAP), ни "сборки мусора", как это предусматривается в АЛГОЛЕ-68. Наконец, сам по себе "C" не обеспечивает никаких возможностей ввода-вывода: здесь нет операторов READ или WRITE и никаких встроенных методов доступа к файлам. Все эти механизмы высокого уровня должны обеспечиваться явно вызываемыми функциями.

 Аналогично, язык "C" предлагает только простые, последовательные конструкции потоков управления: проверки, циклы, группирование и подпрограммы, но не мультипрограммирование, параллельные операции, синхронизацию или сопрограммы. Хотя отсутствие некоторых из этих средств может выглядеть как удручающая неполноценность ("выходит, что я должен обращаться к функции, чтобы сравнить две строки символов ?!"), но удержание языка в скромных размерах дает реальные преимущества. Так как "C" относительно мал, он не требует много места для своего описания и может быть быстро выучен. Компилятор с "C" может быть простым и компактным. Кроме того, компиляторы легко пишутся; при использовании современной технологии можно ожидать написания компилятора для новой ЭВМ за пару месяцев и при этом окажется, что 80 процентов программы нового компилятора будет общей с программой для ужесуществующих компиляторов. Это обеспечивает высокую степень мобильности языка. Поскольку типы данных и структуры управления, имеющиеся в "C", непосредственно поддерживаются большинством существующих ЭВМ, библиотека, необходимая во время прогона изолированных программ, оказывается очень маленькой. На PDP -11, например, она содержит только программы для 32-битового умножения и деления и для выполнения программ ввода и вывода последовательностей. Конечно, каждая реализа-

 ция обеспечивает исчерпывающую, совместимую библиотеку функций для выполнения операций ввода-вывода, обработки строк и распределения памяти, но так как обращение к ним осуществляется только явно, можно , если необходимо, избежать их вызова; эти функции могут быть компактно написаны на самом "C".

 **3.6 Java**

 Сегодня Всемирная сеть - это среда информационного обмена для миллионов людей. Они размещают текст, видео, звук, и информацию, и все более и более, они усложняют свои страницы, делая их интерактивными в сети. JavaScript - это новый язык программирования, используемый в составе страниц HTML для увеличения функциональности и возможностей взаимодействия с пользователями. Он был разработан фирмой Netscape в сотруднечестве с Sun Microsystems на базе языка Sun's Java [1,2].С помощью JavaScript на Web-странице можно сделать то, что невозможно сделать стандартными тегами HTML. Скрипты выполняются в результате наступления каких-либо событий, инициированных действиями пользователя. Создание Web- документов, вклучающих программы на JavaScript, требует наличее текстового редактора и подходящего браузера. Некоторые просмоторщики включают в себе встроенные редакторы, поэтому необходимость во внешнем редакторе отпадает.

 Несмотря на отсутствие прямой связи с языком Java, JavaScript может обращаться к внешним свойствам и методам Java- апплетов, встроенных в страницу HTML. Разница сводится к тому, что апплеты существуют вне браузера, в то время как программы JavaScript могут работать только внутри браузера. На первой взгляд кажется, что найти информацию по JavaScript несложно. Сначала создается впечатление, что ее можно увидеть везде: на сервере Natscape, в виде электронных руководств и примеров, во многих других местах. Тем не менее разыскать информацию об объектах, операторах, цветах и всем прочем в одном источнике, чтобы она была всегда под рукой, трудно.

 **Заключение**

Изобретение языка программирования высшего уровня позволило нам общаться с машиной, понимать её. Развилась наука программирования с того времени, как появились языки программирования, а ведь язык программирования высшего уровня, судя по всему ещё младенец. Но если обратить внимание на темпы роста и развития новейших технологий в области программирования, то можно предположить, что в ближайшем будущем, человеческие познания в этой сфере, помогут произвести на свет языки, умеющие принимать, обрабатывать и передавать информации в виде мысли, слова, звука или жеста. Так и хочется назвать это детище компьютеризированного будущего: «языки программирования "высочайшего" уровня». Возможно, концепция решения этого вопроса проста, а ближайшее будущее этого проекта уже не за горами.

Размышляя над этим, хочется верить в прогресс науки и техники, в высоко - компьютеризированное будущее человечества, как единственного существа на планете, пусть и не использующего один, определенный разговорный язык, но способного так быстро прогрессировать и развивать свой интеллект, что и перехода от многоязыковой системы к всеобщему пониманию долго ждать не придется.

 **Список использованных источников**

1) Родли Джон Создание Java-апплетов .- The Coriolis Group,Inc.,1996, Издательство НИПФ "ДиаСофт Лтд.",1996

2) Эферган Майкл Java: справочник .- QUE Corporation, 1997, Издательство "Питер Ком", 1998

3) Давидов Михаил Изгияевич; Антонов Вадим Геннадьевич “LEX - генератор программ лексического анализа” МОСКВА – 1985;

4) "BASIC Face-off", Justin J.Crom, PC Tech Journal, September 1987, 136 Перевод: Лопухов В.Н. (Интегратор Promt98);

5) Керниган Б.В., Ритчи Д., Фьюэр А. “Язык программирования Си.” Русский перевод: Москва: Финансы и Статистика. 1985 г.;

6) Золотарев В.В., “Основы автоматизации” ч.1, 1978 г.;

7) Ваулин А.С., “Языки программирования” кн.5, 1993 г.;

8) Терренс П. “Языки программирования: разработка и реализация”, 1979 г.;

9) Касвандс Э.Г “Введение в программирование на языке Ассемблер” ч.1;

10) Хротко Г., “Языки программирования высокого уровня”, 1982 г.;

11) Малютин Э.А., Малютина Л.В., “Языки программирования”, 1982 г.;

12) Ушкова В.“Новые языки программирования и тенденции их развития”, 1982 г.;

13) Хьювенен Э., Сеппенен Й., “Мир Лиспа” т.1, 1990 г.;

14) Янг С., “Алгоритмические языки реального времени”, 1985 г.