СОДЕРЖАНИЕ

ВВЕДЕНИЕ **2**

Глава 1. Языки программирования **3**
1.1. История развития языков программирования 10

Глава 2. Обзор современных языков программирования **13**
2.1. Машинно-ориентированные языки программирования (ассемблеры) 14
2.2. Универсальные языки программирования 15
2.3. Процесс создания программы 18

Глава 3. Заключение **20**

ВВЕДЕНИЕ

На современном этапе развития компьютерных технологий невозможно представить какого–либо высококвалифицированного специалиста, не владеющего информационными технологиями. Поскольку деятельность любого субъекта в значительной степени зависит от степени владения информации, а также способности эффективно ее использовать. Для свободной ориентации в информационных потоках современный специалист любого профиля должен уметь получать, обрабатывать и использовать информацию, прежде всего, с помощью компьютеров, а также телекоммуникаций и других новейших средств связи, в том числе и уметь, обращаться с языками программирования.

Актуальность данной темы обусловлена тем, что прогресс компьютерных технологий определил процесс появления новых разнообразных знаковых систем для записи алгоритмов – языков программирования.

Глава 1. Языки программирования

Язык программирования - это система обозначений, служащая для точного описания программ или алгоритмов для ЭВМ. 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. Основные 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 могло бы привести к неправильным ответам при решении задач.

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.

По этому критерию можно выделить следующие уровни языков программирования:

- машинные;
- машинно-ориентированные (ассемблеры);
- машинно-независимые (языки высокого уровня).

Машинные языки и машинно-ориентированные языки — это языки низкого уровня, требующие указания мелких деталей процесса обработки данных. Языки же высокого уровня имитируют естественные языки, используя некоторые слова разговорного языка и общепринятые математические символы. Эти языки более удобны для человека.

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.

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, трудно-обозримой, 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.

Язык ассемблера позволяет программисту пользоваться текстовыми мнемоническими (то есть легко запоминаемыми человеком) 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 процессора. С другой стороны, при этом требуется очень хорошо понимать устройство компьютера, затрудняется отладка больших приложений, а окончательная программа не может быть перенесена на компьютер с другим типом 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 машинной графике, на языке ассемблера пишутся библиотеки, эффективно реализующие алгоритмы обработки изображений, 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 машины. 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 программиста от учета технических особенностей конкретных компьютеров, их архитектуры. 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 ЭВМ. Программа, написанная на таком языке, может выполняться на разных машинах. Составителю программы не нужно знать систему команд ЭВМ, на которой он предполагает проводить вычисления. При

переходе на другую ЭВМ 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 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;
- требуемые операции задаются с помощью общепринятых математических обозначений;
- данным в языках высокого уровня присваиваются индивидуальные имена, выбираемые программистом;
- в языке может быть предусмотрен значительно более широкий набор 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.

Определение 1.1. ***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.

Определение 1.2. ***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.*

Определение 1.3. ***Семантика*** *определяет смысловое значение предложений языка. Являясь системой правил истолкования отдельных языковых конструкций, семантика устанавливает, какие последовательности действий описываются теми или иными фразами языка и, в конечном итоге, какой алгоритм определен данным текстом 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 задачи»: в языке стремятся максимально полно учесть специфику класса задач, для решения которых его предполагается использовать. Например, для многих научно-технических задач характерны большие расчеты по сложным формулам, 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 составления и отладки программы.

Разнообразие классов задач привело к тому, что на сегодняшний день разработано несколько сотен алгоритмических языков. Правда, широкое распространение и международное признание получили лишь 10-15 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.

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.

Объектно-ориентированные языки (Object Pascal, C++, Java, Objective Caml. и др.). Руководящая 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.

Определение 1.4. ***Объект*** *— совокупность свойств (параметров) определенных сущностей и методов их обработки (программных средств).*

Определение 1.5. ***Свойство*** *— это характеристика объекта и его параметров. Все объекты наделены определенными свойствами, совокупность которых выделяют (определяют) объект.*

Определение 1.6. ***Метод*** *— это набор действий над объектом или его свойствами.*

Определение 1.7. ***Событие*** *— это характеристика изменения состояния объекта.*

Определение 1.8. ***Класс*** *— это совокупность объектов, характеризующихся общностью 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 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.

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.

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.

Машинный язык (40-50 годы XX в.).

Программы на машинном языке – очень длинные последовательности единиц и нулей, являлись машинно зависимыми, т.е. для каждой ЭВМ необходимо было составлять свою программу.

Ассемблер (начало 50-ых годов XX в.).

Вместо 1 и 0 программисты теперь могли пользоваться операторами (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).

Первые языки программирования высокого уровня.

С середины 50-ых гг. XX в. начали создавать первые языки программирования высокого уровня (high-level language). Эти языки были Машино независимыми (не привязаны к опред. типу ЭВМ). Но для каждого языка были разработаны собственные компиляторы.

Примеры таких языков: FORTRAN (FORmula TRANslator; 1954) предназначен для научных и технических расчетов; COBOL (1959) был предназначен в

основном для коммерческих приложений (обрабатывал большие объемы нечисловых данных) – Common Business-Oriented Language); язык BASIC (Beginner's All Purpose Instuction Code – универсальный язык символьных инструкций для начинающих) (1964 г.)

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)

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 алгоритм. языка Си;
- язык Object Pascal (1989) был создан на основе языка Pascal. После создания среды программирования – Delphi (1995);
- язык Visual Basic(1991) был создан корпорацией Microsoft на основе языка Qbasic (1975) для разработки приложений с графическим интерфейсом в среде ОС Windows.

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, Mac OS 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 Sun Microsystems для создания сетевого программного обеспечения (1995);
- язык JavaScript – язык сценариев Web-страниц (компания Netscape). (1995)

Языки программирования на платформе .NET.

Интегрированная среда программирования Visual Studio .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 Visual C# (C-шарп) – на основе языков C++ и J – 2003 г.;
- на языке Visual J# (J-шарп) – на основе Java и JavaScript – 2003 г.

Интерпретаторы и компиляторы

Для того, чтобы процессор мог выполнить программу, программа и данные должны быть загружены в оперативную память. Необходимо, чтобы в ОП был 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 исполнимом файле (обычно с расширением .exe).

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 готовой программы – режим компилятора.

Глава 2. Обзор современных языков программирования

Алгоритмический язык (язык программирования) представляет собой один из способов записи алгоритма. Язык программирования является строго формализованным, то есть все команды записываются по определенным правилам и отступления от этих правил не допускаются. 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 FOR I=1 TO 10 считаются правильными, а команды CLEARSCREEN и FOR I FROM 1 TO 10 - неправильными.

Каждая команда, записанная на языке программирования, имеет определенное значение, то есть заставляет компьютер выполнять те или иные действия. 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. Языки отличаются друг от друга множеством допустимых объектов и набором операций, которые можно выполнять над этими объектами.

Программа, написанная на языке программирования, представляет собой просто текст. Чтобы компьютер мог выполнять команды, содержащиеся в этой программе, надо перевести программу в набор понятных компьютеру инструкций, записанных в двоичной форме (в 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.

2.1. Машинно-ориентированные языки программирования (ассемблеры)

Первым значительным шагом представляется переход к языку ассемблера. Не очень заметный, казалось бы, шаг — переход к символическому кодированию машинных команд — имел на самом деле огромное значение.

Программисту не надо было больше вникать в хитроумные способы кодирования команд на аппаратном уровне. Более того, зачастую одинаковые по сути команды кодировались совершенно различным образом в зависимости от своих параметров.

Появилась также возможность использования макросов и меток, что также упрощало создание, модификацию и отладку программ. Появилось даже некое подобие переносимости — существовала возможность разработки целого семейства машин со сходной системой команд и некоего общего ассемблера для них, при этом не было нужды обеспечивать двоичную совместимость.

Вместе с тем, переход к новому языку таил в себе и некоторые отрицательные (по крайней мере, на первый взгляд) стороны. Становилось почти невозможным использование всяческих хитроумных приемов сродни тем, что упомянуты выше.

Кроме того, здесь впервые в истории развития программирования появились два представления программы: в исходных текстах и в откомпилированном виде. Сначала, пока ассемблеры только транслировали мнемоники в машинные

коды, одно легко переводилось в другое и обратно, но затем по мере появления таких возможностей, как метки и макросы, дизассемблирование становилось все более и более трудным делом. К концу ассемблерной эры возможность автоматической трансляции в обе стороны была утеряна окончательно. В связи с этим было разработано большой количество специальных программ-дизассемблеров, осуществляющих обратное преобразования, однако в большинстве случаев они с трудом могут разделить код и данные. Кроме того, вся логическая информация (имена переменных, меток и т.п.) теряется безвозвратно. В случае же задачи о декомпиляции языков высокого уровня примеры удовлетворительного решения проблемы и вовсе единичны.

Каждый оператор языка представляет собой мнемоническое (условное) обозначение машинной команды. Естественно, что каждый тип процессора имеет свой набор команд, а значит, свой ассемблер. Ассемблеры используются 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.

2.2. Универсальные языки программирования

Иногда их 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.

Среди универсальных языков программирования в настоящее время наиболее распространены:

Си его разновидности

- Си [C] - Многоцелевой язык программирования высокого уровня, разработанный Денисом Ритчи в начале 1970-х гг. на базе языка BCPL. Используется на мини ЭВМ и ПЭВМ. Является базовым языком операционной системы Unix, однако применяется и вне этой системы, для написания быстродействующих и эффективных программных продуктов, включая и операционные системы. Для IBM PC имеется ряд популярных версий языка Си, в том числе - Turbo C (фирмы Borland), Microsoft C и Quick

C (фирмы Microsoft), а также Zortech C (фирмы Symantec). Многие из указанных версий обеспечивают также работу с Си и Си++.

- Си++ [C++] - Язык программирования высокого уровня, созданный Бьярном Страуструпом 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].

Паскаль

Паскаль [PASCAL - акроним с французского - Program Applique a la Selection et la 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.

Фортран

В 1954 году в недрах корпорации IBM группой разработчиков во главе с Джоном Бэкусом (John Backus) был создан язык программирования Fortran.

Значение этого события трудно переоценить. Это первый язык программирования высокого уровня. Впервые программист мог по-настоящему абстрагироваться от особенностей машинной архитектуры. Ключевой идеей, отличающей новый язык от ассемблера, была концепция подпрограмм. Напомним, что это современные компьютеры поддерживают подпрограммы на аппаратном уровне, предоставляя соответствующие команды и структуры данных (стек) прямо на уровне ассемблера, в 1954 же году это было совершенно не так. Поэтому компиляция Fortran'а была процессом отнюдь не тривиальным. Кроме того, синтаксическая структура языка была достаточно сложна для машинной обработки в первую очередь из-за того, что пробелы как синтаксические единицы вообще не использовались. Это порождало массу возможностей для скрытых ошибок, таких, например: в Фортране следующая конструкция описывает "цикл for до метки 10 при изменении индекса от 1 до 100": DO 10 I=1,100. Если же здесь заменить запятую на точку, то получится оператор присваивания: DO10I = 1.100 Говорят, что такая ошибка заставила ракету взорваться во время старта.

Язык Фортран использовался (и используется по сей день) для научных вычислений. Он страдает от отсутствия многих привычных языковых конструкций и атрибутов, компилятор практически никак не проверяет синтаксически правильную программу с точки зрения семантической корректности (соответствие типов и проч.). В нем нет поддержки современных способов структурирования кода и данных. Это осознавали и сами разработчики. По признанию самого Бэкуса, перед ними стояла задача скорее разработки компилятора, чем языка. Понимание самостоятельного значения языков программирования пришло позже.

Появление Фортрана было встречено еще более яростной критикой, чем внедрение ассемблера. Программистов пугало снижение эффективности программ за счет использования промежуточного звена в виде компилятора. И эти опасения имели под собой основания: действительно, хороший программист, скорее всего, при решении какой-либо небольшой задачи вручную напишет код, работающий быстрее, чем код, полученный как результат компиляции. Через некоторое время пришло понимание того, что реализация больших проектов невозможна без применения языков высокого уровня. Мощность вычислительных машин росла, и с тем падением эффективности, которое раньше считалось угрожающим, стало возможным смириться. Преимущества же языков высокого уровня стали настолько очевидными, что побудили разработчиков к созданию новых языков, все более и более совершенных.

Бейсик

Бейсик [BASIC - Beginner's All-purpose Symbolic Instruction Code] - 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.

Например, язык Lisp используется для создания экспертных систем. Язык Java используется для разработки сетевых (Web)- приложений.

2.3. Процесс создания программы

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 заданном листе 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 C есть среда Turbo C и Borland C.

Глава 3. Заключение

Изобретение языков программирования высшего уровня, а также их постоянное совершенствование и развитие, позволило человеку не только общаться с машиной и понимать ее, но использовать ЭВМ для сложнейших расчетов в области самолетостроения, ракетостроения, медицины и даже экономики.

На сегодняшний день, любое среднее и крупное предприятие, имеет в своем штате группу программистов, обладающими знаниями программирования различными языками, которые редактируют, изменяют, и модифицируют программы используемыми сотрудниками предприятия. Это говорит о том, что на рынке труда пользуются спросом обладающими знаниями и опытом работы с различными языками программирования.

В данной курсовой работе, нами были рассмотрены самые распространенные языки программирования, такие как: Фортран, Паскаль, Бейсик, которые используется для научных вычислений, для обучения программированию начинающих программистов.

Несмотря на то, что современный уровень развития языков программирования находятся на высоком уровне, тенденция их развития, а также развития информационных технологий в целом, складывается таким образом, что 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.