Содержание:

ВВЕДЕНИЕ

Языки программирования, тема данной контрольной работы, в которой мы постараемся разобраться в различных языках и их использовании. Данная тема выбрана не случайно так как, в настоящее время программисты, пишущие программы и многое другое, очень востребованы на рынке труда.

Практически в любой области необходимо программировать различные системы оборудования и т.д. Значимость программирования трудно переоценить в настоящее время.

Со временем создания первых программируемых машин человечество придумало огромное количество языков программирования их более 8 тысяч.

В данной курсовой работе мы познакомимся с практически самыми первыми языками программирования. Не затронем только “Машинный язык” который стал родоначальником всех языков программирования.

Рассмотрим и познакомимся с языком программирования: Ассемблер, FORTRAN, COBOL, ALGOL и BASIC. Узнаем их историю, познакомимся с особенностями, что было позаимствовано из других языков и улучшено или разработано самостоятельно, а также чем прославился и запомнился каждый из языков программирования.

1. Язык программирования: Ассемблер

Первая программа на компьютере, написанная в машинных кодах, была запущена в 1948 году в Англии. Довольно скоро стало понятно, что процесс формирования машинного кода можно автоматизировать.

Уже в 1950 году для записи программ начали применять мнемонический язык – язык Assembler. Язык ассемблера позволил представить машинный код в более удобной для человека форме: для обозначения команд и объектов, над которыми эти команды выполняются, вместо двоичных кодов использовались буквы или

сокращенные слова, которые отражали суть команды.

Исторически, если первым поколением языков программирования считать машинные коды, то язык ассемблера можно рассматривать как второе поколение языков программирования.

Ассемблер является языком программирования низкого уровня и предназначен для машинного программирования.

Низкоуровневый язык программирования имеет наилучшую совместимость с машинным кодом что позволяет программировать каждый этап.

Ассемблер – машинно-ориентировочный язык программирования низкого уровня. Его команды прямо соответствуют отдельным программам машины или их последовательностям, также он может представлять дополнительные возможности облегчения программирования, такие как макрокоманды, выражения, средства обеспечения модульности программ. Может рассматривается как “автокод” расширенных конструкциями языков программирования высокого уровня.

Автокод - это язык программирования, предложения которого по своей структуре в основном подобны командам и обрабатываемым данным конкретного машинного языка.

Язык ассемблера – это система обозначений, используемая для представления в удобно читаемой форме программ, записанных в машинном коде. Язык ассемблера позволяет программисту пользоваться алфавитными мнемоническими кодами операций, по своему усмотрению присваивать символические имена регистрам ЭВМ и памяти, а также задавать удобные для себя схемы адресации (например, индексную или косвенную). Кроме того, он позволяет использовать различные системы счисления (например, десятичную или шестнадцатеричную) для представления числовых констант и даёт возможность помечать строки программы метками с символическими именами с тем, чтобы к ним можно было обращаться (по именам, а не по адресам) из других частей программы (например, для передачи управления)

Команды языка ассемблера один к одному соответствуют командам процессора.

Фактически, они представляют собой более удобную для человека символьную форму записи - мнемокоды - команд и их аргументов. При этом одной команде

языка ассемблера может соответствовать несколько вариантов команд процессора.

Кроме того, язык ассемблера позволяет использовать символические метки вместо адресов ячеек памяти, которые при ассемблировании заменяются на вычисляемые ассемблером или компоновщиком абсолютные, или относительные адреса, а также так называемые директивы (команды ассемблера, не переводимые в машинные команды процессора, а выполняемые самим ассемблером).

Директивы ассемблера позволяют: включать блоки данных, задать ассемблирование фрагмента программы по условию, задать значения меток, использовать макрокоманды с параметрами.

Каждая модель (или семейство) процессоров имеет свой набор - систему - команд и соответствующий ему язык ассемблера. Наиболее популярные синтаксисы языков ассемблера - Intel-синтаксис и AT&T-синтаксис.

Использование языка ассемблера предоставляет программисту ряд возможностей, как правило, недоступных при программировании на языках высокого уровня. Большинство из них связано с близостью языка к аппаратной платформе. За счёт более рационального использования ресурсов процессора, например, максимально эффективного размещения всех исходных данных в регистрах, можно исключить излишние обращения к оперативной памяти; а также за счёт ручной оптимизации вычислений, в том числе более эффективного использования промежуточных результатов, может быть сокращён объём кода и повышена скорость программы.

Недостатками языка ассемблер можно считать сложность разработки на нём больших программных комплексов что и привело к появлению языков третьего поколения - языков программирования высокого уровня (таких как Фортран, Лисп, Кобол, Паскаль, Си и др.). Именно языки программирования высокого уровня и их наследники в основном используются в настоящее время в индустрии информационных технологий. Однако языки ассемблера сохраняют свою нишу, обусловленную их уникальными преимуществами в части эффективности и возможности полного использования специфических средств конкретной платформы.

Возможность максимально полного использования всех особенностей аппаратной платформы позволяет, теоретически, писать самый быстрый и компактный код из возможных для данного процессора. Искусный программист, как правило, способен значительно оптимизировать программу по сравнению с транслятором с языка высокого уровня по одному или нескольким параметрам и создать код, близкий к

оптимальному по Парето

Использование ассемблера практически не имеет альтернативы при создании: драйверов оборудования и ядра операционной системы (по крайней мере, машинно-зависимых подсистем ядра ОС), тогда, когда важно временное согласование работы периферийных устройств с центральным процессором. Программ, которые должны храниться в ПЗУ ограниченного объёма и/или выполняться на устройствах с ограниченной производительностью (прошивок компьютеров и различных электронных устройств)

Отдельно можно отметить, что с помощью программы-дизассемблера возможно преобразование откомпилированной программы в программу на языке ассемблера.

2. Язык программирования: FORTRAN

Fortran - первый язык программирования высокого уровня, получивший практическое применение

Высокоуровневый язык программирования проще для восприятия человеком в отличии от низкоуровнего языка программирования. Данный язык больше похож на обычный текст, который поймет любой человек, и тем проще на нем разрабатывать какие-либо программы

В конце 1953 Джон Бэкус предложил начать разработку эффективной альтернативы ассемблеру для программирования на ПК IBM 704. Уже к середине 1954 была закончена черновая спецификация языка Fortran.

В период с 1954 по 1957 года был разработан и выпущен группой программистов под руководством Джона Бэкуса в корпорации IBM Название Fortran является сокращением от “FORmula TRANslator”.

Язык был широко принят учеными для написания программ с интенсивными вычислениями. Включение комплексного типа данных сделало его особенно подходящим для технических приложений. Но в то же время сообщество относилось скептически к новому способу программирования и не верили в то, что Fortran позволит программировать быстрее и эффективнее.

К 1960 году существовали версии Fortran для компьютеров IBM 709, 650, 1620, 7090. Его большая популярность побуждала конкурирующих изготовителей

компьютеров создавать компиляторы Fortran для своих компьютеров. Таким образом, уже к 1963 существовало более 40 компиляторов для разных платформ. Именно по этому Fortran считают первым широко используемым языком программирования.

Эволюция:

1. Первым стал FORTRAN IV
2. На основе первого создан FORTRAN 66 (1966)
3. FORTRAN 77 (1978) получил множество улучшений: Строковый тип данных и функции для его обработки, Блочные операторы IF,ELSE IF, ELSE, END IF, операторы включения фрагмента программы INCLUDE и т.д.
4. В FORTRAN 90 (1991) значительно переработан стандарт языка. Введён свободный формат написания кода. Появились дополнительные описания IMPLICIT NONE, TYPE, ALLOCATABLE, POINTER, TARGET, NAMELIST; управляющие конструкции DO … END DO, DO WHILE, CYCLE, SELECT CASE, WHERE; работа с динамической памятью (ALLOCATE, DEALLOCATE, NULLIFY); программные компоненты MODULE, PRIVATE, PUBLIC, CONTAINS, INTERFACE, USE, INTENT. Появились новые встроенные функции, в первую очередь, для работы с массивами. В языке появились элементы ООП
5. FORTRAN 95 (1997) получил коррекция предыдущего стандарта.
6. FORTRAN 2003 (2004) получил дальнейшее развитие поддержки ООП в языке. Взаимодействие с операционной системой
7. FORTRAN 2008 (2010) Стандартом предполагается поддержка средствами языка параллельных вычислений (Co-Arrays Fortran). Также предполагается увеличить максимальную размерность массивов до 15, добавить встроенные специальные математические функции и др.
8. FORTRAN 2018 Последняя версия

Имеется большое количество написанных на Фортране различных математических библиотек для матричной алгебры и решения систем линейных уравнений, библиотеки для решения дифференциальных уравнений, интегральных уравнений и их систем, аппроксимации функций, специальных функций, быстрых преобразований Фурье, математической статистики и других математических дисциплин. Эти библиотеки поставляются, как правило, с компилятором. Ряд таких пакетов создавался на протяжении десятилетий и популярен в научной среде по сей день, например – IMSL

Большинство таких библиотек является фактически достоянием человечества: они доступны в исходных кодах, хорошо документированы, отлажены и весьма эффективны.

Fortran имеет достаточно большой набор встроенных математических функций, поддерживает работу с целыми, вещественными и комплексными числами высокой точности. Выразительные средства языка изначально были весьма бедны, поскольку Fortran был одним из первых языков высокого уровня. В дальнейшем были добавлены многие лексические конструкции, характерные для структурного, функционального и даже объектно-ориентированного программирования.

Структура программ изначально была ориентирована на ввод с перфокарт и имела ряд удобных именно для этого случая свойств. Так, 1-я колонка служила для маркировки текста как комментария (символом C), с 1-й по 5-ю располагалась область меток, а с 7-й по 72-ю располагался собственно текст оператора или комментария. Колонки с 73-й по 80-ю могли служить для нумерации карт (чтобы восстановить случайно рассыпавшуюся колоду) или для краткого комментария, транслятором они игнорировались. Если текст оператора не вписывался в отведённое пространство (с 7-й по 72-ю колонку), в 6-ой колонке следующей карты ставился признак продолжения, и затем оператор продолжался на ней. Расположить два или более оператора в одной строке (карте) было нельзя. Когда перфокарты ушли в историю, эти достоинства превратились в серьёзные неудобства.

Именно поэтому в стандарте, начиная с Fortran 90, в добавление к фиксированному формату исходного текста появился свободный формат, который не регламентирует позиции строки, а также позволяет записывать более одного оператора на строку. Введение свободного формата позволило создавать код, читабельность и ясность.

Многие системы программирования позволяют компоновать полученные в результате трансляции программы на Fortran объектные файлы с объектными файлами, полученными от компиляторов с других языков, что позволяет создавать более гибкие и многофункциональные приложения.

В Фортране вызов подпрограмм, функций и передача их параметров происходят исключительно по ссылке, а не по значению. Поэтому подпрограмма может изменить переданный ей аргумент в главной программе, если специальным образом это не предотвратить. Такой механизм позволяет сделать естественной

нотацию при записи математических формул и сохранить при этом высокое быстродействие при работе с большими массивами данных.

Стандарт языка позволяет осуществлять перегрузку процедур и операций посредством родового интерфейса, объединяя различные процедуры под одним именем. В этом случае в главной программе достаточно обратиться к родовой процедуре, а характер выполняемых операций будет зависеть от типа данных, предложенных процедуре для обработки. По такому принципу сделаны все встроенные функции и подпрограммы

3. Язык программирования: COBOL

COBOL (от COmmon Business-Oriented Language) - один из старейших языков программирования, разработанный прежде всего для написания программ для экономической сферы.

Спецификация языка была создана в 1959 году. Создатели языка ставили своей целью сделать его машиннонезависимым и максимально приближенным к естественному английскому языку. Обе цели были успешно достигнуты; программы на COBOL считаются понятными даже неспециалистам, поскольку тексты на этом языке программирования не нуждаются в каких-либо специальных комментариях.

COBOL - язык очень старый и в свое время использовался крайне активно, поэтому существует множество реализаций и диалектов. Для языка был утвержден ряд стандартов: в 1968, 1974, 1985 и 2002 годах. Последний стандарт добавил в язык поддержку объектно-ориентированной парадигмы.

Язык позволяет эффективно работать с большим количеством данных, он насыщен разнообразными возможностями поиска, сортировки и распределения. К числу других плюсов COBOL обычно относят его структурированность. Довольно мощные компиляторы с этого языка разработаны для персональных компьютеров. Некоторые из них столь эффективны, что программу, отлаженную на персональном компьютере, нетрудно перенести на большие ЭВМ.

Основной минус заключается в том, что на Коболе можно запрограммировать лишь простейшие алгебраические вычисления. Для сложных инженерных расчетов этот язык не годится.

4. Язык программирования: ALGOL

Algol - название ряда языков программирования, применяемых при составлении программ для решения научно-технических задач на ЭВМ. Разработан комитетом по языку высокого уровня IFIP в 1958 – 1960 годах

После принятия в 1958 году первой версии описания языка Algol 58 довольно быстро были осознаны проблемы, для решения которых комитет сформировал новый вариант стандарта - Algol 60; он и стал “классическим” Алголом.

Особенности языка Алгол стали типичными для большинства императивных языков, созданных позднее него. Именно в Алголе появилось представление о программе не как о свободной последовательности команд, а как о блочной структуре, состоящей из чётко описанных и отделённых друг от друга частей. Основной блок программы на Алголе - это сама главная программа. Она содержит свою исполняемую часть, заключённую в блок, ограниченный парой ключевых слов begin и end, а также описания подпрограмм. Каждая подпрограмма - это программа в миниатюре, имеющая собственные, описанные внутри неё данные, однозначно определённый интерфейс в виде имени и списка формальных параметров, и блок кода. При этом в блоке могут выделяться подблоки.

Крайне важным свойством Алгола стала возможность организации рекурсивных процедур, до этого у промышленных языков отсутствовавшая

5. Язык программирования: BASIC

Данный язык программирования был разработан в 1964 году профессорами Дартмутского колледжа Томасом Курцем и Джоном Кемени.

Предпосылкой к появлению Бейсика стали повышение доступности компьютеров в 1960-х годах и появление возможности работы в режиме разделения времени. К компьютерам получили доступ учащиеся и специалисты, не являющиеся подготовленными программистами, но нуждающиеся в решении на компьютере своих задач.

Бейсик был спроектирован так, чтобы студенты могли без затруднений писать программы, используя терминалы с разделением времени. Он предназначался для более “простых” пользователей, не столько заинтересованных в скорости

исполнения программ, сколько просто в возможности использовать компьютер для решения своих задач, не имея специальной подготовки.

Язык был основан частично на Фортране II и частично на Алголе 60, с добавлениями, делающими его удобным для работы в режиме разделения времени и, позднее, обработки текста и матричной арифметики. Первоначально Бейсик был реализован на мейнфрейме GE-265 с поддержкой множества терминалов.

Изначально язык уже использовался на нескольких мини-компьютерах, его настоящее распространение началось с его появления на микрокомпьютере Altair 8800. Многие языки программирования были слишком большими, чтобы поместиться в небольшую память, которую пользователи таких машин могли себе позволить. Для машин с таким медленным носителем как бумажная лента без подходящего текстового редактора такой небольшой язык, как Бейсик, был отличной находкой.

Внешний вид программ на ранних версиях Бейсика во многом определялся тем, что он предназначался для среды программирования со строчным редактором текста. В таком редакторе пользователь не имел возможности отображать весь текст на экране, перемещаться по нему в любых направлениях с помощью клавиатуры и мыши. В строчных редакторах пользователь для изменения строки текста должен дать команду изменения строки с заданным номером, затем ввести новый текст указанной строки. Для вставки новой строки нужно было дописать строку с номером, находящимся в диапазоне между номерами двух других строк. Если требовалось вставить новую строку между строками с соседними номерами, приходилось повторно вводить эти строки с другими номерами, чтобы обеспечить разрыв в нумерации либо применять специальную команду “RENUM”, которая перенумеровывала все уже введённые строки с постоянным шагом, синхронно изменяя все команды переходов. Изменять вручную номера уже имеющихся строк было небезопасно, так как номера выполняли роль меток для оператора GOTO.

Ранее в Бейсике были допустимы только числовые метки и наличие в каждой строке уникального номера делало возможным легкий поиск и переход к любой из нужных строк. Числовые метки и активное использование GOTO было неизбежным при программировании на Бейсике, что способствовало плохой структуре кода и в больших проектах приводило к запутыванию самих авторов.

Периодом наибольшего расцвета и развития исходного Бейсика можно считать конец 1970-х - первую половину 1980-х годов. В этот период практически все

персональные компьютеры имели собственный интерпретатор Бейсика, зачастую зашиваемый в ПЗУ, который в простейших компьютерах нередко выполнял также функции операционной системы. Практически все более мощные компьютеры также имели интерактивные Бейсик-системы. Характерно, что практически не было компиляторов Бейсика, абсолютное большинство реализаций представляли собой стандартный гибрид строчного редактора и интерпретатора. Недовольство примитивизмом исходного Бейсика порождало попытки улучшить и базовые средства языка, что привело к появлению в некоторых реализациях сложных операторов ветвления, дополнительных видов циклов, именованных процедур с параметрами.

Было создано несколько новых версий Бейсика для платформы IBM PC. Microsoft продавала Бейсик для MS-DOS/PC DOS, включая IBM Advanced BASIC (BASICA), GW-BASIC (модификация BASICA, не требующая «прошивки» от IBM) и QuickBASIC.

В некоторых диалектах Бейсика имелась возможность загрузить файл с кодом непосредственно при выполнении программы. Это была одна из самых полезных возможностей языка, так как она позволяла разбить большой проект на модули, вызывая их в порядке необходимости.

Редактора кода в ранних версиях, как такового, не было. При запуске интерпретатора Бейсика запускался диалоговый режим ввода команд. Для работы в этом режиме были предусмотрены специальные команды, которые не являлись операторами самого языка. Точка с запятой выполняет функцию подавления перехода на новую строку после оператора PRINT. Ключевое слово END означает завершение программы. Оно необязательно, в его отсутствие программа завершалась после исполнения последнего оператора, но было полезно в случаях, когда имелась секция подпрограмм, либо программа имела несколько возможных точек завершения. Запуск программы производился вводом команды RUN. В большинстве интерпретаторов Бейсика в нижней части экрана была строка с этими командами.

Начиная с конца 80-х, новые компьютеры стали намного более сложными и предоставляли возможности (такие как графический интерфейс пользователя), которые делали Бейсик уже не столь удобным для программирования. Бейсик начал сдавать свои позиции, несмотря на то, что огромное количество его версий ещё использовалось и продавалось.

Вторую жизнь Бейсик получил с появлением в 1991 году Visual Basic от Microsoft. Этот язык напоминал оригинальный только синтаксисом, но был значительно более современным. Visual Basic и его варианты стали одним из наиболее часто используемых языков на платформе Windows. Позже был создан вариант под названием WordBasic, используемый в MS Word до появления Word 97. Visual Basic for Applications (VBA) был встроен в Excel 5.0 в 1993 году, затем в Access 95 в 1995-ом, а после и во все остальные инструменты, входящие в пакет Office — в 1997-ом. Internet Explorer 3.0 и выше, а также поставки Windows включали интерпретатор скриптового языка VBScript. В полный вариант пакета OpenOffice также включён интерпретатор Бейсика.

Практически все недостатки, присущие ранним версиям языка, были исправлены, и программирование на современных диалектах Бейсика мало отличается от использования других языков и сред.

Оборотной стороной процесса модификации Бейсик является то, что язык стал гораздо объёмнее и сложнее, его освоение требует больше времени и усилий.

ЗАКЛЮЧЕНИЕ

Мы познакомились с пятью языками программирования, узнали их историю и особенности

Трудно конечно определить, какой язык программирования наиболее популярен и востребован в данный момент и какой будет позднее. Один из языков может отнимать наибольшее количество человеко-часов, на другом написано наибольшее число строк кода, третий занимает наибольшее процессорное время, а четвёртый наиболее часто служит исследовательской базой в академических кругах. Каждый язык в своем роде очень популярны для конкретных задач.

Язык программирования: Ассемблер является низкоуровневым и подходить для работы на машинном уровне. Использовать его по другому назначению можно, НО это не целесообразно так как займет много времени.

Для других целей лучше подойдут высокоуровневые языки программирования, в которых количество опций больше и лучше, а так же не требует большого количества времени

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Программирование на ассемблере на платформе x86-64 / Р. Аблязов. - М.: Книга по Требованию, 2011. - 302 с.

2. Информатика: учебник / Б.В. Соболь, А.Б. Галин, Ю.В. Панов и др. - Ростов н/Д: Феникс, 2012. -- 446 с.

3. Программирование на Алголе. / В.М. Савинков, В.Д. Цальп. 1975г.

4. Горелик А.М Эволюция языка программирования Фортран (1957—2007) и перспективы его развития // Вычислительные методы и программирование, 2008, Т. 9, с. 53-71

5. Себастьян Рашка Python Machine Learning, September 23, 2015, 417 с.

“Эволюция языков программирования”

6. ГОСТ 23056-78 Язык программирования Фортран

7. ГОСТ 22558-89 Язык программирования КОБОЛ