МБОУ «СОШ №23»

Самостоятельная творческая работа

Языки программирования

Автор: *ученица 5 класса Морозова Валерия*

Руководитель: *Дорожинская И.Ф.*

Сергиев Посад
2015 год

С каждым годом компьютеры получают все большее распространение. Они становятся быстрее и удобнее в использовании, а профессия программиста уже давно стала одной и самых востребованных и высокооплачиваемых. Даже далекий от **программирования** человек почти наверняка слышал о том, что существуют **языки программирования**. Для чего же они нужны и почему их так много?

Подробнее: http://www.kakprosto.ru/kak-89609-dlya-chego-nuzhny-yazyki-programmirovaniya#ixzz3U9nZvs1M

Каким бы совершенным ни был компьютер, без программного обеспечения он представляет собой просто груду металла и пластика. Именно программы определяют, что и как делает компьютер, в какой последовательности он выполняет те или иные операции. Первые **языки программирования** начали появляться в начале пятидесятых годов и использовались для преобразования простых арифметических выражений в машинный код. Машинный код – это система команд вычислительной машины, интерпретируемых непосредственно микропроцессором. Но человеку писать программу в машинных кодах очень неудобно. Для того чтобы облегчить труд программиста, и начали создаваться **языки программирования**. Языки **программирования** делятся на **языки** высокого уровня и низкого. Чем выше уровень языка, тем легче на нем писать программисту. Такой язык более понятен человеку, так как позволяет с помощью простых смысловых конструкций задавать необходимую последовательность действий. После создания программы происходит ее компиляция – то есть автоматический перевод в понятный процессору язык машинных кодов. Языки низкого уровня находятся гораздо ближе к языку машинных кодов, поэтому писать на них труднее. Но у них есть свое преимущество – написанные на таком языке программы получаются очень быстрыми и компактными. Наиболее популярным низкоуровневым языком является Assembler. Некоторые его преимущества настолько очевидны, что даже в сложных программах, написанных на высокоуровневых языках, часто применяют вставки на Ассемблере. Несмотря на существование большого количества языков **программирования**, получившие широкое распространение можно пересчитать по пальцам. Одним из самых распространенных является язык C++. Это очень удобный и достаточно простой для программиста язык, позволяющий создавать программы любого уровня сложности. Не так давно компания Microsoft разработала язык C# (читается как «си шарп»), обладающий рядом новых возможностей и предназначенный для написания программ под операционную систему Windows. Компания Microsoft выпустила и очень популярную среду **программирования** Microsoft Visual Studio, позволяющую программировать на C++, C# и некоторых других языках. Очень известным является язык **программирования** Delphi. Свое происхождение он ведет от некогда знаменитого Паскаля, однако благодаря усилиям компании Borland приобрел ряд новых качеств, став, по сути, новым языком. Писать на этом языке достаточно просто и удобно, а благодаря среде **программирования** Borland Delphi он получил очень широкое распространение. Без языков **программирования** было бы невозможным и существование интернета. Такие **языки**, как Perl и PHP позволяют создавать скрипты, определяющие выполнение на страницах сайта необходимых действий. Даже создание самой простой интернет-страницы невозможно без знания HTML – стандартного языка разметки документов. Вычислительные устройства сейчас находятся повсюду: в сотовых телефонах и банкоматах, в станках с числовым программным управлением и в телевизорах. Трудно найти сферу жизни, в которой они не были бы тем или иным образом задействованы. И все эти устройства работают благодаря программам, написанным с помощью тех или иных языков **программирования**.

Подробнее: http://www.kakprosto.ru/kak-89609-dlya-chego-nuzhny-yazyki-programmirovaniya#ixzz3U9nZvs1M

Содержание:

Введение

До середины 60-х компьютеры были слишком дорогими машинами, использовавшимися только для особых задач, и выполнявшими только одну задачу за раз (т. н. пакетная обработка).

Языки программирования этой эры, как и компьютеры на которых они использовались, были разработаны для специфичных задач, таких как научные вычисления. Поскольку машины были дорогими и лишь одна задача выполнялась за раз, то и машинное время было дорого – поэтому скорость выполнения программы стояла на первом месте.

Однако в течение 60-х цена на компьютеры стала падать так, что даже небольшие компании могли их себе позволить; скорость компьютеров всё увеличивалась и наступило время, когда они стали часто простаивать без задач. Чтобы этого не происходило, стали вводить системы с разделением времени (time-sharing).

В таких системах процессорное время «нарезалось», и все пользователи поочерёдно получали короткие отрезки этого времени. Машины были достаточно быстрыми для того, чтобы в результате каждый пользователь за терминалом чувствовал себя так, будто работает с системой в одиночку. Машина же, в свою очередь, простаивала меньше, поскольку выполняла не одну, а сразу много задач. Разделение времени радикально снижало стоимость машинного времени, поскольку одна машина могла совместно использоваться сотнями пользователей.

В этих условиях - когда мощность стала дешева и доступна - создатели языков программирования все больше стали задумываться об удобстве написания программ, а не только скорости их выполнения. «Мелкие» (атомарные) операции, выполняемые непосредственно устройствами машины, объединили в более «крупные», высокоуровневые операции и целые конструкции, с которыми человеку куда проще и удобнее работать.

Что такое язык программирования

Язык программирования — формальная знаковая система, предназначенная для описания алгоритмов в форме, которая удобна для исполнителя (например, компьютера). Язык программирования определяет набор лексических, синтаксических и семантических правил, используемых при составлении компьютерной программы. Он позволяет программисту точно определить то, на какие события будет реагировать компьютер, как будут храниться и передаваться данные, а также какие именно действия следует выполнять над этими при различных обстоятельствах.

Со времени создания первых программируемых машин человечество придумало уже более двух с половиной тысяч языков программирования. Каждый год их число пополняется новыми. Некоторыми языками умеет пользоваться только небольшое число их собственных разработчиков, другие становятся известны миллионам людей. Профессиональные программисты иногда применяют в своей работе более десятка разнообразных языков программирования.

Создатели языков по-разному толкуют понятие язык программирования. Среди общин мест, признаваемых большинством разработчиков, находятся следующие:

Функция: язык программирования предназначен для написания компьютерных программ, которые применяются для передачи компьютеру инструкций по выполнению того или иного вычислительного процесса и организации управления отдельными устройствами.

Задача: язык программирования отличается от естественных языков тем, что предназначен для передачи команд и данных от человека компьютеру, в то время как естественные языки используются лишь для общения людей между собой. В принципе,

можно обобщить определение "языков программирования" - это способ передачи команд, приказов, чёткого руководства к действию; тогда как человеческие языки служат также для обмена информацией.

Исполнение: язык программирования может использовать специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений.

Для чего нужны языки программирования

Процесс работы компьютера заключается в выполнении программы, то есть набора вполне определённых команд во вполне определённом порядке. Машинный вид команды, состоящий из нулей и единиц, указывает, какое именно действие должен выполнить центральный процессор. Значит, чтобы задать компьютеру последовательность действий, которые он должен выполнить, нужно задать последовательность двоичных кодов соответствующих команд. Программы в машинных кодах состоят из тысячи команд. Писать такие программы – занятие сложное и утомительное. Программист должен помнить комбинацию нулей и единиц двоичного кода каждой программы, а также двоичные коды адресов данных, используемых при её выполнении. Гораздо проще написать программу на каком-нибудь языке, более близком к естественному человеческому языку, а работу по переводу этой программы в машинные коды поручить компьютеру. Так возникли языки, предназначенные специально для написания программ, **- языки программирования.**

Имеется много различных языков программирования. Вообще-то для решения большинства задач можно использовать любой из них. Опытные программисты знают, какой язык лучше использовать для решения каждой конкретной задачи, так как каждый из языков имеет свои возможности, ориентацию на определённые типы задач, свой способ описания понятий и объектов, используемых при решении задач.

Всё множество языков программирования можно разделить на две группы: **языки низкого уровня** и **языки высокого уровня.**

К языкам низкого уровня относятся языки ассемблера (от англ. to assemble – собирать, компоновать). В языке ассемблера используются символьные обозначения команд, которые легко понятны и быстро запоминаются. Вместо последовательности двоичных кодов команд записываются их символьные обозначения, а вместо двоичных адресов данных, используемых при выполнении команды, - символьные имена этих данных, выбранные программистом. Иногда язык ассемблера называют мнемокодом или автокодом.

Большинство программистов пользуются для составления программ языками высокого уровня. Как и обычный человеческий язык, такой язык имеет свой алфавит – множество символов, используемых в языке. Из этих символов составляются так называемые ключевые слова языка. Каждое из ключевых слов выполняет свою функцию, так же как в привычном нам языке нам языке слова, составленные из букв алфавита данного языка, могут выполнять функции разных частей речи. Ключевые слова связываются друг с другом в предложения по определённым синтаксическим правилам языка. Каждое предложение определяет некоторую последовательность действий, которые должен выполнить компьютер.

Язык высокого уровня выполняет роль посредника между человеком и компьютером, позволяя человеку общаться с компьютером более привычным для человека способом. Часто такой язык помогает выбрать правильный метод решения задачи.

Перед тем как писать программу на языке высокого уровня, программист должен составить алгоритм решения задачи, то есть пошаговый план действий, который нужно выполнить для решения этой задачи. Поэтому языки, требующие предварительного составления алгоритма, часто называют алгоритмическими языками.

Какие существуют языки программирования

Фортран

Языки программирования стали появляться уже с середины 50-х годов. Одним из первых языков такого типа стал язык Фортран (англ. FORTRAN от FORmula TRANslator – переводчик формул), разработанный в 1957 году. Фортран применяется для описания алгоритма решения научно-технических задач с помощью ЦВМ. Так же, как и первые вычислительные машины, этот язык предназначался, в основном, для проведения естественно-научных и математических расчётов. В усовершенствованном виде этот язык сохранился до нашего времени. Среди современных языков высокого уровня он является одним из наиболее используемых при проведении научных исследований. Наиболее распространены варианты Фортран-II, Фортран-IV, EASIC Fortran и их обобщения.

Алгол

После Фортрана в 1958-1960 годах появился язык Алгол (Алгол-58, Алгол-60) (англ. ALGOL от ALGOrithmic Language – алгоритмический язык). Алгол был усовершенствован в 1964-1968 годах – Алгол-68. Алгол был разработан комитетом, в который входили европейские и американские учёные. Он относится к языкам высокого уровня (high-level language) и позволяет легко переводить алгебраические формулы в программные команды. Алгол был популярен в Европе, в том числе СССР, в то время как сравнимый с ним Фортран был распространен в США и Канаде. Алгол оказал заметное влияние на все разработанные позднее языки программирования, и, в частности, на язык Pascal. Этот язык так же, как и Фортран, предназначался для решения научно-технических задач. Кроме того, этот язык применялся как средство обучения основам программирования – искусства составления программ.

Обычно под понятием Алгол подразумевается язык Алгол-60, в то время как Алгол-68 рассматривается как самостоятельный язык. Даже когда язык Алгол почти перестал использоваться для программирования, он ещё оставался официальным языком для публикации алгоритмов.

Кобол

В 1959 – 1960 годах был разработан язык Кобол (англ. COBOL от COmmom Business Oriented Language – общий язык, ориентированный на бизнес). Это язык

программирования третьего поколения, предназначенный, в первую очередь, для разработки бизнес приложений. Также Кобол предназначался для решения экономических задач, обработки данных для банков, страховых компаний и других учреждений подобного рода. Разработчиком первого единого стандарта Кобола являлась Грейс Хоппер (бабушка Кобола).

Кобол обычно критикуется за многословность и громоздкость, поскольку одной из целей создателей языка было максимально приблизить конструкции к английскому языку. (До сих пор Кобол считается языком программирования, на котором было написано больше всего строк кода). В то же время, Кобол имел прекрасные для своего времени средства для работы со структурами данных и файлами, что обеспечило ему долгую жизнь в бизнес приложениях, по крайней мере, в США.

Лисп

Почти одновременно с Коболом (1959 – 1960 гг.) в Массачусетском технологическом институте был создан язык Лисп (англ. LISP от LISt Processing – обработка списков). Лисп основан на представлении программы системой линейных списков символов, которые притом являются основной структурой данных языка. Лисп считается вторым после Фортрана старейшим высокоуровневым языком программирования. Этот язык широко используется для обработки символьной информации и применяется для создания программного обеспечения, имитирующего деятельность человеческого мозга.

Любая программа на Лиспе состоит из последовательности выражений (форм). Результат работы программы состоит в вычислении этих выражений. Все выражения записываются в виде списков — одной из основных структур Лиспа, поэтому они могут легко быть созданы посредством самого языка. Это позволяет создавать программы, изменяющие другие программы или макросы, позволяющие существенно расширить возможности языка.

Основной смысл Лисп-программы "жизнь" в символьном пространстве: перемещение, творчество, запоминание, создание новых миров и т.д. Лисп как метафора мозга, символ, метафора сигнала: "Как происходит биологический анализ сигналов мозгом, как внешний фактор - физическое и химическое воздействие, являющееся для организма раздражителем превращается в биологически значимый сигнал, зачастую жизненно важный, определяющий все поведение человека или животного; и как происходит разделение разных сигналов на положительные, отрицательные и безразличные, индифферентные. Сигнал это уже интегративное понятие. Он представляет собой опознавательный знак группы, комплексных раздражителей, связанных между собой общей историей и причинно следственными отношениями. В этом комплексе, системе раздражителей, сигнальный стимул сам является также составляющим элементом и при иных обстоятельствах его роль может принадлежать другому стимулу из комплекса. В сигнале концентрируется весь прошлый опыт животного или человека."[1]

Бейсик

В середине 60-х годов (1963 г.) в Дартмутском колледже (США) был создан язык Бейсик (англ. BASIC от Beginner’s Allpurpose Instruction Code – всецелевой символический код

инструкций для начинающих). Со временем, когда стали появляться другие диалекты, этот «изначальный» диалект стали называть Dartmouth BASIC. Язык был основан частично на Фортран II и частично на Алгол-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. Вопреки распространённому убеждению, в момент своего появления это был компилируемый язык.

Бейсик был спроектирован так, чтобы студенты могли писать программы, используя терминалы с разделением времени. Он создавался как решение для проблем, связанных со сложностью более старых языков. Он предназначался для более «простых» пользователей, не столько заинтересованных в скорости программ, сколько просто в возможности использовать компьютер для решения своих задач. В силу простоты языка Бейсик многие начинающие программисты начинают с него свой путь в программировании.

Форт

В конце 60-х – начале 70-х годов появился язык Форт (англ. FOURTH – четвёртый). Этот язык стал применяться в задачах управления различными системами после того, как его автор Чарльз Мур написал на нём программу, предназначенную для управления радиотелескопом Аризонской обсерватории.

Ряд свойств, а именно интерактивность, гибкость и простота разработки делают Форт весьма привлекательным и эффективным языком в прикладных исследованиях и при создании инструментальных средств. Очевидными областями применения этого языка являются встраиваемые системы управления. Также находит применение при программировании компьютеров под управлением различных операционных систем.

Паскаль

Появившийся в 1972 году язык Паскаль был назван так в честь великого французского математика XVII века, изобретателя первой в мире арифметической машины Блеза Паскаля. Этот язык был создан швейцарским учёным, специалистом в области информатики Никлаусом Виртом как язык для обучения методам программирования. Паскаль – это язык программирования общего назначения.

Особенностями языка являются строгая типизация и наличие средств структурного (процедурного) программирования. Паскаль был одним из первых таких языков. По мнению Н. Вирта, язык должен способствовать дисциплинированию программирования, поэтому, наряду со строгой типизацией, в Паскале сведены к минимуму возможные синтаксические неоднозначности, а сам синтаксис интуитивно понятен даже при первом знакомстве с языком.

Язык Паскаль учит не только тому, как правильно написать программу, но и тому, как правильно разработать метод решения задачи, подобрать способы представления и организации данных, используемых в задаче. С 1983 года язык Паскаль введён в учебные курсы информатики средних школ США.

Ада

На основе языка Паскаль в конце 70-х годов был создан язык Ада, названный в честь одарённого математика Ады Лавлейс (Огасты Ады Байрон – дочери поэта Байрона). Именно она в 1843 году смогла объяснить миру возможности Аналитической машины Чарльза Бэббиджа. Язык Ада был разработан по заказу Министерства обороны США и первоначально предназначался для решения задач управления космическими полётами. Этот язык применяется в задачах управления бортовыми системами космических кораблей, системами обеспечения жизнедеятельности космонавтов в полёте, сложными техническими процессами.

Ада — это структурный, модульный, объектно-ориентированный язык программирования, содержащий высокоуровневые средства программирования параллельных процессов. Синтаксис Ады унаследован от языков типа Algol или Паскаль, но расширен, а также сделан более строгим и логичным. Ада — язык со строгой типизацией, в нём исключена работа с объектами, не имеющими типов, а автоматические преобразования типов сведены к абсолютному минимуму.

По утверждению Стефена Цейглера[2], разработка программного обеспечения на Аде в целом обходится на 60 % дешевле, а разработанная программа имеет в 9 раз меньше дефектов, чем при использовании языка Си.

Си

В настоящее время популярным среди программистов является язык Си (С – буква английского алфавита). Язык Си берёт своё начало от двух языков - BCPL и B. В 1967 году Мартин Ричардс разработал BCPL как язык для написания системного программного обеспечения и компиляторов. В 1970 году Кен Томпсон использовал B для создания ранних версий операционной системы UNIX на компьютере DEC PDP-7. Как в BCPL, так и в B переменные не разделялись на типы - каждое значение данных занимало одно слово в памяти и ответственность на различение, например, целых и действительных чисел целиком ложилась на плечи программиста.

Язык Си был разработан (на основе B) Деннисом Ритчи из Bell Laboratories и впервые был реализован в 1972 году на компьютере DEC PDP-11. Известность Си получил в качестве языка ОС UNIX. Сегодня практически все основные операционные системы были написаны на Си или C++. По прошествии двух десятилетий Си имеется в наличии на большинстве компьютеров. Он не зависит от аппаратной части.

В конце 70-х годов Си превратился в то, что мы называем «традиционный Си». В 1983 году Американским комитетом национальных стандартов в области компьютеров и обработки информации был учрежден единый стандарт этого языка.

Этот язык имеет богатые средства, позволяет писать гибкие программы, использующие все возможности современных персональных компьютеров.

Пролог

Ещё один язык, который считается языком будущего, был создан в начале 70-х годов группой специалистов Марсельского университета. Это язык Пролог. Своё название он получил от слов «ПРОграммирование на языке ЛОГики». В основе этого языка лежат законы математической логики. Как и язык Лисп, Пролог применяется, в основном, при проведении исследований в области программной имитации деятельности мозга человека. В отличие от описанных выше языков, этот язык не является алгоритмическим. Он относится к так называемым **дескриптивным** (от англ. descriptive – описательный) – описательным языкам. Дескриптивный язык не требует от программиста разработки всех этапов выполнения задачи. Вместо этого, в соответствии с правилами такого языка, программист должен описать базу данных, соответствующую решаемой задаче, и набор вопросов, на которые нужно получить ответы, используя данные из этой базы.

В последние десятилетия в программировании возник и получил существенное развитие **объектно-ориентированный** подход. Это метод программирования, имитирующий реальную картину мира: информация, используемая для решения задачи, представляется в виде множества взаимодействующих объектов. Каждый из объектов имеет свои свойства и способы поведения. Взаимодействие объектов осуществляется при помощи передачи сообщений: каждый объект может получать сообщения от других объектов, запоминать информацию и обрабатывать её определённым способом и, в свою очередь, посылать сообщения. Так же, как и в реальном мире, объекты хранят свои свойства и поведение вместе, наследуя часть из них от родительских объектов.

Объектно-ориентированная идеология используется во всех современных программных продуктах, включая операционные системы.

Первый объектно-ориентированный язык Simula-67 был создан как средство моделирования работы различных приборов и механизмов. Большинство современных языков программирования – объектно-ориентированные. Среди них последние версии языка Turbo-Pascal, C++, Ada и другие.

В настоящее время широко используются системы **визуального программирования** Visual Basic, Visual C++, Delphi и другие. Они позволяют создавать сложные прикладные пакеты, обладающие простым и удобным пользовательским интерфейсом.

Что такое компилятор и интерпретатор

Создать язык, удобный для написания программ, недостаточно. Для каждого языка нужен свой переводчик. Такими переводчиками являются специальные программы-трансляторы.

Транслятор – это программа, предназначенная для перевода программы, написанной на одном языке программирования, в программу на другом языке программирования. Процесс перевода называется трансляцией.

Тексты исходной и результирующей программ находятся в памяти компьютера.

Примером транслятора является компилятор.

Компилятор – это программа, предназначенная для перевода программы, написанной на каком-либо языке, в программу в машинных кодах. Процесс такого перевода называется компиляцией.

Компилятор создаёт законченный результат – программу в машинных кодах. Затем эта программа выполняется. Откомпилированный вариант исходной программы можно сохранить на диске. Для повторного выполнения исходной программы компилятор уже не нужен. Достаточно загрузить с диска в память компьютера откомпилированный в предыдущий раз вариант и выполнить его.

Существует другой способ сочетания процессов трансляции и выполнения программы. Он называется интерпретацией. Суть процесса интерпретации состоит в следующем. Вначале переводится в машинные коды, а затем выполняется первая строка программы. Когда выполнение первой строки окончено, начинается перевод второй строки, которая затем выполняется и так далее. Управляет этим процессом программа-интерпретатор.

Интерпретатор – это программа, предназначенная для построчных трансляции и выполнения исходной программы. Такой процесс называется интерпретацией.

В процесс трансляции входит проверка исходной программы на соответствие правилам используемого в ней языка. Если в программе обнаружены ошибки, транслятор вводит сообщение о них на устройство вывода (обычно, на экран дисплея).

Интерпретатор сообщает о найденных им ошибках после трансляции каждой строки программы. Это значительно облегчает процесс поиска и исправления ошибок в программе, однако существенно увеличивает время трансляции. Компилятор транслирует программу намного быстрее, чем интерпретатор, но сообщает о найденных им ошибках после завершения компиляции всей программы. Найти и исправить ошибки в этом случае труднее. Поэтому интерпретаторы рассчитаны, в основном, на языки, предназначенные для обучения программированию, и используются начинающими программистами. Большинство современных языков предназначены для разработки сложных пакетов программ и рассчитаны на компиляцию.

Иногда один и тот же язык может использовать и компилятор, и интерпретатор. К числу таких языков относится, например, Бейсик.

Как правило, программы-компиляторы и интерпретаторы называются так же, как и языки, для перевода с которых они предназначены. Слова Паскаль, Ада, Си могут относиться как к названиям языков, так и к названиям соответствующих программ.