ГОУ ВПО «Государственный Медицинский Университет»

Кафедра Информатики и вычислительной техники

Реферат по информатике на тему

«Языки программирования»

Выполнил: Ларин С.Л.

Проверил: Артеменко

Андрей Эдуардович

Курск, 2006

**План**

Введение

Стандартизация языков программирования

Типы данных

Структуры данных

Семантика языков программирования

Процедурные языки программирования

Basic

Pascal

Си

Объектно-ориентированные языки программирования

Си++

JAVA

Заключение

Литература

**Введение**

Мнения о том, какой язык лучше преподавать, разнятся: от того, что программирование изучать не нужно, а следует просто поднимать компьютерную грамотность и осваивать офисные программы (как на Западе), до того, что нужно изучать операционные системы и несколько языков программирования различных уровней абстракции и с различными парадигмами. Это крайние случаи, но золотую середину найти непросто. В первую очередь, нам нужно определить цель. Научить логически и алгоритмически мыслить? Познакомить с компьютерами на бытовом уровне, чтобы Вы умели пользоваться интернетом, электропочтой и текстовыми редакторами? Заложить базовые знания, необходимые для будущих инженеров, математиков, физиков и специалистов по информационным технологиям? А может, нам нужно каждого познакомить с программированием как явлением, чтобы он представлял потенциал компьютерных систем? Много ли из нас станет программистами? Немного. Но синусами и уравнениями Кирхгофа в жизни тоже пользуется не каждый. Безусловно, в науке о программировании есть фундаментальная составляющая, но определить её не просто. Некоторые считают, что не так важно, какой язык программирования взять: на уроках информатики нужно учить не языку программирования, а методам программирования и системному подходу решения задач. Нужно развивать алгоритмическое мышление и на примерах знакомиться с принципами построения современных компьютерных систем. Неужели действительно не так важно, какая среда и какой конкретный язык программирования будет использован для практических занятий? Оказывается, что у каждого преподавателя есть свой список требований к учебному языку программирования. Например: простой, интуитивный синтаксис, наличие высокоуровневых инструментов для обнаружения и недопущения ошибок и для отладки программ, наличие качественной документации с примерами, наличие дружелюбной среды разработки, межплатформенность (наличие версий под различные платформы). У некоторых преподавателей этот список очень короткий, например: «Только Паскаль» или «Любой, кроме Бейсика!»

**Стандартизация языков программирования**

Язык программирования может быть представлен в виде набора спецификаций, определяющих его синтаксис и семантику.

Для многих широко распространённых языков программирования созданы международные стандарты. Специальные организации проводят регулярное обновление и публикацию спецификаций и формальных определений соответствующего языка. В рамках таких комитетов продолжается разработка и модернизация языков программирования и решаются вопросы о расширении или поддержке уже существующих и новых языковых конструкций.

**Типы данных**

Современные цифровые компьютеры обычно являются двоичными и данные хранят в двоичном (бинарном) коде (хотя возможны реализации и в других системах счисления). Эти данные как правило отражают информацию из реального мира (имена, банковские счета, измерения и др.), представляющую высокоуровневые концепции.

Особая система, по которой данные организуются в программе, — это *система типов* языка программирования; разработка и изучение систем типов известна под названием теория типов. Языки могут быть классифицированы как системы *со статической типизацией* и языки с *динамической типизацией*.

Статически-типизированные языки могут быть в дальнейшем подразделены на языки с *обязательной декларацией*, где каждая переменная и объявление функции имеет обязательное объявление типа, и языки с *выводимыми типами*. Иногда динамически-типизированные языки называются *латентно-типизированными*.

**Структуры данных**

Системы типов в языках высокого уровня позволяют определять сложные, составные типы, так называемые структуры данных. Как правило, структурные типы данных образуются как декартово произведение базовых (атомарных) типов и ранее определённых составных типов.

Основные структуры данных (списки, очереди, хеш-таблицы, двоичные деревья и пары) часто представлены особыми синтаксическими конструкциями в языках высокого уровня. Такие данные структурируются автоматически.

**Семантика языков программирования**

Существует несколько подходов к определению семантики языков программирования.

Наиболее широко распространены разновидности следующих трёх: операционного, денотационного (математического) и деривационного (аксиоматического).

При описании семантики в рамках операционного подхода обычно исполнение конструкций языка программирования интерпретируется с помощью некоторой воображаемой (абстрактной) ЭВМ.

Деривационная семантика описывает последствия выполнения конструкций языка с помощью языка логики и задания пред- и постусловий. Денотационная семантика оперирует понятиями, типичными для математики — множества, соответствия, а также суждения, утверждения и др.

Язык программирования строится в соответствии с той или иной базовой моделью вычислений и парадигмой программирования.

Несмотря на то, что большинство языков ориентировано на императивную модель вычислений, задаваемую фоннеймановской архитектурой ЭВМ, существуют и другие подходы. Можно упомянуть языки со стековой вычислительной моделью (Forth, Factor, Postscript и др.), а также функциональное (Лисп, Haskell, ML и др.) и логическое программирование (Пролог) и язык Рефал, основанный на модели вычислений, введённой советским математиком А.А. Марковым-младшим.

В настоящее время также активно развиваются проблемно-ориентированные, декларативные и визуальные языки программирования.

**Компилируемые и интерпретируемые языки**

Языки программирования могут быть разделены на компилируемые и интерпретируемые.

Программа на компилируемом языке при помощи специальной программы компилятора преобразуется (компилируется) в набор инструкций для данного типа процессора (машинный код) и далее записывается в исполнимый модуль, который может быть запущен на выполнение как отдельная программа. Другими словами, компилятор переводит исходный текст программы с языка программирования высокого уровня в двоичные коды инструкций процессора.

Если программа написана на интерпретируемом языке, то интерпретатор непосредственно выполняет (интерпретирует) исходный текст без предварительного перевода. При этом программа остаётся на исходном языке и не может быть запущена без интерпретатора. Можно сказать, что процессор компьютера — это интерпретатор машинного кода.

Кратко говоря, компилятор переводит исходный текст программы на машинный язык сразу и целиком, создавая при этом отдельную исполняемую программу, а интерпретатор выполняет исходный текст прямо во время исполнения программы.

Разделение на компилируемые и интерпретируемые языки является несколько условным. Так, для любого традиционно компилируемого языка, как, например, Паскаль, можно написать интерпретатор. Кроме того, большинство современных «чистых» интерпретаторов не исполняют конструкции языка непосредственно, а компилируют их в некоторое высокоуровневое промежуточное представление (например, с разыменованием переменных и раскрытием макросов).

Для любого интерпретируемого языка можно создать компилятор — например, язык Лисп, изначально интерпретируемый, может компилироваться без каких бы то ни было ограничений. Создаваемый во время исполнения программы код может так же динамически компилироваться во время исполнения.

Как правило, скомпилированные программы выполняются быстрее и не требуют для выполнения дополнительных программ, так как уже переведены на машинный язык. Вместе с тем, при каждом изменении текста программы требуется её перекомпиляция, что создаёт трудности при разработке. Кроме того, скомпилированная программа может выполняться только на том же типе компьютеров и, как правило, под той же операционной системой, на которую был рассчитан компилятор. Чтобы создать исполняемый файл для машины другого типа, требуется новая компиляция.

Интерпретируемые языки обладают некоторыми специфическими дополнительными возможностями (см. выше), кроме того, программы на них можно запускать сразу же после изменения, что облегчает разработку. Программа на интерпретируемом языке может быть зачастую запущена на разных типах машин и операционных систем без дополнительных усилий.

Однако интерпретируемые программы выполняются заметно медленнее, чем компилируемые, кроме того, они не могут выполняться без дополнительной программы-интерпретатора.

Некоторые языки, например, Java и C#, находятся между компилируемыми и интерпретируемыми. А именно, программа компилируется не в машинный язык, а в машинно-независимый код низкого уровня, байт-код. Далее байт-код выполняется виртуальной машиной. Для выполнения байт-кода обычно используется интерпретация, хотя отдельные его части для ускорения работы программы могут быть транслированы в машинный код непосредственно во время выполнения программы по технологии компиляции «на лету» (Just-in-time compilation, JIT). Для Java байт-код исполняется виртуальной машиной Java (Java Virtual Machine, JVM), для C# — Common Language Runtime.

Подобный подход в некотором смысле позволяет использовать плюсы как интерпретаторов, так и компиляторов. Следует упомянуть также оригинальный язык Форт(Forth) имеющий и интерпретатор и компилятор.

Современные языки программирования рассчитаны на использование ASCII, то есть доступность всех *графических* символов ASCII является необходимым и достаточным условием для записи любых конструкций языка. *Управляющие* символы ASCII используются ограниченно: допускаются только возврат каретки CR, перевод строки LF и горизонтальная табуляция HT (иногда также вертикальная табуляция VT и переход к следующей странице FF).

*Подробнее по этой теме см.: Переносимый набор символов.*

Ранние языки, возникшие в эпоху 6-битных символов, использовали более ограниченный набор. Например, алфавит Фортрана включает 49 символов (включая пробел): A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9 = + - \* / () . , $ ' :

Заметным исключением является язык APL, в котором используется очень много специальных символов.

Использование символов за пределами ASCII (например, символов KOI8-R или символов Юникода) зависит от реализации: иногда они разрешаются только в комментариях и символьных/строковых константах, а иногда и в идентификаторах. В СССР существовали языки, где все ключевые слова писались русскими буквами, но большую популярность подобные языки не завоевали (исключение составляет. Встроенный язык программирования 1С: Предприятие).

*Подробнее по этой теме см.: Русские языки программирования.*

Расширение набора используемых символов сдерживается тем, что многие проекты по разработке программного обеспечения являются международными. Очень сложно было бы работать с кодом, где имена одних переменных записаны русскими буквами, других — арабскими, а третьих — китайскими иероглифами. Вместе с тем, для работы с текстовыми данными языки программирования нового поколения (Delphi 2006, C#, Java) поддерживают Unicode.

**Процедурные языки программирования**

Процедурное (императивное) программирование является отражением архитектуры традиционных ЭВМ, которая была предложенафон Нейманом в 1940-х годах. Теоретической моделью процедурного программирования служит алгоритмическая система под названием Машина Тьюринга.

Программа на процедурном языке программирования состоит из последовательности операторов (инструкций), задающих процедуру решения задачи. Основным является оператор присваивания, служащий для изменения содержимого областей памяти. Концепция памяти как хранилища значений, содержимое которого может обновляться операторами программы, является фундаментальной в императивном программировании.

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

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

**Basic**

Бейсик (от BASIC, сокращение от англ. *Beginner’s All-purpose Symbolic Instruction Code* — *универсальный код символических инструкций для начинающих*; англ. *basic*— *основной*, *базовый*) — семейство высокоуровневых языков программирования.

Был разработан в 1963 году профессорами Дартмутского колледжа Томасом Куртом (*Thomas E. Kurtz*) и Джоном Кемени (*John G. Kemeny*).

Язык предназначался для обучения программированию и получил широкое распространение в виде различных диалектов, прежде всего, как язык для домашних микрокомпьютеров.

*Предпосылки*

До середины 1960-х компьютеры (ЭВМ) были слишком дорогими машинами, использовавшимися в основном для научно-технических задач, и выполнявшими задачи не в интерактивном режиме (т.н. *пакетная обработка*), когда общение с машиной осуществлялось через её оператора.

Языки программирования этой поры, как и компьютеры, на которых они использовались, были разработаны для работы с ними обученных технических специалистов и программистов. Поскольку машины были дорогими, то и машинное время было дорого — поэтому на первом месте стояла скорость выполнения программы (скорость счёта).

Однако, в течение 1960-х цены на компьютеры стали падать так, что даже небольшие компании могли их себе позволить; их быстродействие всё увеличивалась и наступило время, когда стало возможно организовать более эффективную непосредственную работу пользователей с компьютерами в так называемом режиме разделения времени (*time-sharing*).

*Рождение*

Бейсик был придуман в 1963 году преподавателями Дартмутского Колледжа Джоном Кемени и Томасом Куртцом, и под их руководством был реализован командой студентов колледжа. Со временем, когда стали появляться другие диалекты, этот «изначальный» диалект стали называть Dartmouth BASIC.

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.

При проектировании языка использовались следующие восемь принципов: новый язык должен

* быть простым в использовании *для начинающих*
* быть языком программирования *общего назначения*
* предоставлять возможность *расширения функциональности*, доступную опытным программистам
* быть *интерактивным*
* предоставлять ясные сообщения об ошибках
* быстро работать на небольших программах
* не требовать понимания работы аппаратного обеспечения
* защищать пользователя от операционной системы

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.

*Взрывной рост*

Несмотря на то что язык уже использовался на нескольких миникомпьютерах, 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 (позднее — аудиокассета) и без подходящего текстового редактора такой небольшой язык как Бейсик был отличной находкой. В 1975 году Майкрософт (тогда это были лишь двое — Билл Гейтс и Пол Аллен, при участии Монте Давидова) выпустила Altair BASIC. Затем его версии появились на другой платформе под лицензией и скоро в использовании были уже миллионы копий и вариантов; один из вариантов, Applesoft BASIC, стал стандартным языком на Apple II. Для операционной системы CP/M был создан диалект BASIC-80, надолго определивший развитие языка.

В 1979 году Майкрософт обсуждала с несколькими поставщиками компьютеров (включая IBM) лицензирование интерпретатора Бейсик на их машинах. Одна из версий (ROM BASIC) была включена в ПЗУ IBM PC — компьютер мог автоматически загружаться в Бейсик. Так как IBM не придавала тогда большого значения персональным компьютерам (основным полем её деятельности были мейнфреймы), то, вопреки своим принципам, она разрешила Microsoft продавать интерпретатор отдельно. Это сделало последнюю пионером в выпуске ПО нового поколения — не привязанного к конкретной аппаратуре и поставляемого отдельно от компьютера.

Синтаксис языка напоминает Fortran, и многие элементы — явные заимствования из него. Язык задумывался для обучения, поэтому его конструкции максимально просты. Как и в других языках программирования, ключевые слова взяты из английского языка. Основных типов данных два: строки и числа. С появлением версии Visual Basic, а также различных его модификаций (таких как VBA), в языке появились многие другие типы данных и дополнения, типичные для современных языков программирования (например, такие как объектная модель). Объявление переменных не требует специальной секции (в отличие, например, от Паскаля). Объявление переменной — это первое её использование.

Ранние версии бейсика (такие как GWBASIC, QBasic) существенно отличаются от современных диалектов и на сегодня практически не используются.

**Pascal**

Паскаль (англ. *Pascal*) — высокоуровневый язык программирования общего назначения. Один из наиболее известных языков программирования, широко применяется в промышленном программировании, обучении программированию в высшей школе, является базой для большого числа других языков. Был создан Никлаусом Виртом в 1968/9 годах (опубликован в 1970-м) после его участия в работе комитета разработки стандарта языка Алгол-68.

Паскаль был создан как язык для обучения процедурному программированию (хотя, по словам Вирта, язык нельзя считать только учебным, поскольку язык, непригодный для написания реальных программ, для обучения использоваться не должен). Название языку дано в честь выдающегося французского математика, физика, литератора и философа Блеза Паскаля.

Паскаль был создан Никлаусом Виртом в 1968/9 годах после его участия в работе комитета разработки стандарта языка Алгол-68. Он был опубликован в 1970 году Виртом как небольшой и эффективный язык, чтобы способствовать хорошему стилю программирования, использовать структурное программирование и структурированные данные.

Компилятор Паскаля был написан на самом Паскале, используя «метод раскрутки», когда создается ядро языка, с постепенным наращиванием новых возможностей.

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 автор постарался сделать интуитивно понятным даже при первом знакомстве с языком.

Тем не менее, первоначально язык имел ряд ограничений: невозможность передачи функциям массивов переменной длины, отсутствие нормальных средств работы с динамической памятью, ограниченная библиотека ввода-вывода, отсутствие средств для подключения функций написанных на других языках, отсутствие средств раздельной компиляции и т.п. Подробный разбор недостатков языка Паскаль того времени был выполнен Брайаном Керниганом в статье «Почему Паскаль не является моим любимым языком программирования» (интересно, что эта статья вышла в начале 1980-х, когда уже существовал язык Модула-2, потомок Паскаля, избавленный от большинства его пороков, а также более развитые диалекты Паскаля). Некоторые недостатки Паскаля были исправлены в ISO-стандарте 1982 года, в частности, в языке появились открытые массивы, давшие возможность использовать одни и те же процедуры для обработки одномерных массивов различных размеров.

Необходимо заметить, что многие недостатки языка не проявляются или даже становятся достоинствами при обучении программированию. Кроме того, по сравнению с основным языком программирования в академической среде 70-х (которым былФортран, обладавший гораздо более существенными недостатками), Паскаль представлял собой значительный шаг вперёд. В начале 1980-х годов в СССР для обучения школьников основам информатики и вычислительной техники академик А.П. Ершов разработал алголо-паскалеподобный «алгоритмический язык».

Наиболее известной реализацией Паскаля, обеспечившая широкое распространение и развитие языка является Turbo Pascal фирмы Borland, выросшая затем в объектный Паскаль для DOS (начиная с версии 5.5) и Windows и далее в Delphi, в которой использовались значительные расширения языка.

Диалекты Паскаля, применяемые в Turbo Pascal для DOS и Delphi для Windows, из-за отстутствия других успешных коммерческих реализаций стали популярны.

Объектный Паскаль от Borland и в отсутствие отраслевой стандартизации стал сильно отличаться от классического Паскаля.

Язык программирования — формальная знаковая система, предназначенная для записи программ, задающих алгоритм в форме, понятной 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 данными при различных обстоятельствах. Со времени создания первых программируемых машин человечество придумало уже более восьми 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.

**Cи**

Си (англ. *C*) — стандартизированный процедурный язык программирования, разработанный в начале 1970-х годов сотрудниками Bell Labs Кеном Томпсоном и Денисом Ритчи как развитие языка Би. Си был создан для использования в операционной системе UNIX. С тех пор он был импортирован на многие другие операционные системы и стал одним из самых используемых языков программирования. Си ценят за его эффективность; он является самым популярным языком для создания системного программного обеспечения. Его также часто используют для создания прикладных программ. Несмотря на то, что Си не разрабатывался для новичков, он активно используется для обучения программированию. В дальнейшем синтаксис языка Си стал основой для многих других языков (см.: Си-подобный синтаксис).

Для языка Си характерны лаконичность, современный набор конструкций управления потоком выполнения, структур данных и обширный набор операций.

Язык программирования Си отличается минимализмом. Авторы языка хотели, чтобы программы на нём легко компилировались с помощью однопроходного компилятора, после компиляции каждой элементарной составляющей программы соответствовало весьма небольшое число машинных команд, а использование базовых элементов языка не задействовало библиотеку времени выполнения. Однопроходный компилятор компилирует программу, не возвращаясь назад, к уже откомпилированному тексту. Поэтому использованию функции должно предшествовать её объявление. Код на Си можно легко писать на низком уровне абстракции, почти как на ассемблере. Иногда Си называют «универсальным ассемблером» или «ассемблером высокого уровня», что отражает различие языков ассемблера для разных платформ и единство стандарта Си, код которого может быть скомпилирован без изменений практически на любой модели компьютера. Си часто называют языком *среднего уровня* или даже *низкого уровня*, учитывая то, как близко он работает к реальным устройствам.

Компиляторы Си разрабатываются сравнительно легко благодаря относительно низкому уровню языка и скромному набору элементов. Поэтому данный язык доступен на самых различных платформах (возможно, круг этих платформ шире, чем у любого другого существующего языка). К тому же, несмотря на свою низкоуровневую природу, язык позволяет создавать переносимые программы и поддерживает программиста в этом. Программы, соответствующие стандарту языка, могут компилироваться на самых различных компьютерах.

Си (как и ОС UNIX, с которой он долгое время был связан) создавался программистами и для программистов, круг которых был бы ненамного шире круга разработчиков языка. Несмотря на это, область использования языка значительно шире задач системного программирования.

Си создавался с одной важной целью: сделать более простым написание больших программ с минимумом ошибок по правилам процедурного программирования, не добавляя лишних накладных расходов на итоговый код программы компилятором, как это всегда делают языки очень высокого уровня, такие как Бейсик. С этой стороны Си имеет следующие важные особенности:

* простую языковую базу, из которой вынесены в библиотеки многие существенные возможности, вроде математических функций или функций управления файлами;
* ориентацию на процедурное программирование, обеспечивающую удобство применения структурного стиля программирования;
* систему типов, предохраняющую от бессмысленных операций;
* использование препроцессора для, например, определения макросов и включения файлов с исходным кодом;
* непосредственный доступ к памяти компьютера через использование указателей;
* минимальное число ключевых слов;
* передачу параметров в функцию по значению, а не по ссылке (при этом передача по ссылке выполняется с помощью указателей);
* указатели на функции и статические переменные
* области действия имён;
* записи — определяемые пользователем собирательные типы данных (структуры), которыми можно манипулировать как одним целым;

Вот некоторые особенности других языков программирования, которых не имеет Си:

* автоматическое управление памятью;
* поддержка объектно-ориентированного программирования (при этом первые версии C++ генерировали код программы на языке Си);
* вложенные функции (существуют компиляторы языка Си реализующие эту функцию, например компилятор GNU);
* полиморфизм функций и операторов;
* встроенная поддержка многозадачности и сети
* функции высшего порядка
* карринг.

Несмотря на то, что в Си нет столь многого, а это было важно в начале, язык был хорошо принят, потому что он позволял быстро создавать компиляторы для новых платформ, а также позволял программистам довольно точно представлять, как выполняются их программы. Благодаря этому программы, написанные на Си, эффективнее написанных на многих других языках. Как правило, лишь оптимизированный вручную код на ассемблере может работать ещё быстрее, потому что он даёт полный контроль над машиной, однако развитие современных компиляторов вместе с усложнением современных процессоров быстро сократило этот разрыв.

Одним из последствий высокой эффективности и переносимости Си стало то, что многие компиляторы, интерпретаторы и библиотеки других языков высокого уровня часто выполнены на языке Си.

**Объектно-ориентированные языки программирования**

Объектно-ориентированный язык программирования (ОО-язык) — язык, построенный на принципах объектно-ориентированного программирования.

В основе концепции объектно-ориентированного программирования лежит понятие *объекта* - некоей субстанции, которая объединяет в себе поля (данные) и методы (выполняемые объектом действия).

Например, объект "человек" может иметь поля "имя", "фамилия" и иметь методы "есть" и "спать".Соответственно, мы можем использовать в программе операторы Человек. Имя: ="Иван" и Человек. Есть(пища).

**Си++**

Компилируемый статически типизированный язык программирования общего назначения. Поддерживает разные парадигмы программирования, но, в сравнении с его предшественником — языком Си, — наибольшее внимание уделено поддержкеобъектно-ориентированного и обобщённого программирования.[1]

Название «Си++» происходит от Си, в котором унарный оператор ++ обозначает инкрементпеременной.

В 1990-х годах язык стал одним из наиболее широко применяемых языков программирования общего назначения.

При создании Си++ стремились сохранить совместимость с языком Си. Большинство программ на Си будут исправно работать и с компилятором Си++. Си++ имеет синтаксис, основанный на синтаксисе Си.

Язык возник в начале 1980-х годов, когда сотрудник фирмы Bell Laboratories Бьёрн Страуструп придумал ряд усовершенствований к языку Си под собственные нужды. До начала официальной стандартизации язык развивался в основном силами Страуструпа в ответ на запросы программистского сообщества. В 1998 году был ратифицирован международный стандарт языка Си++: ISO/IEC 14882:1998 «Standard for the C++ Programming Language»; после принятия технических исправлений к стандарту в 2003 году — нынешняя версия этого стандарта — ISO/IEC 14882:2003. Ранние версии языка, известные под именем «C с классами», начали появляться с 1980 года. Идея создания нового языка берёт начало от опыта программирования Страуструпа для диссертации. Он обнаружил, что язык моделирования Симула (Simula) имеет такие возможности, которые были бы очень полезны для разработки большого программного обеспечения, но работает слишком медленно. В то же время язык BCPL достаточно быстр, но слишком близок к языкам низкого уровня и не подходит для разработки большого программного обеспечения. Страуструп начал работать в Bell Labs над задачами теории очередей (в приложении к моделированию телефонных вызовов). Попытки применения существующих в то время языков моделирования оказались неэффективными. Вспоминая опыт своей диссертации, Страуструп решил дополнить язык Си (преемник BCPL) возможностями, имеющимися в языке Симула. Язык Си, будучи базовым языком системы UNIX, на которой работали компьютеры Bell, является быстрым, многофункциональным и переносимым. Страуструп добавил к нему возможность работы с классами и объектами. В результате, практические задачи моделирования оказались доступными для решения как с точки зрения времени разработки (благодаря использованию Симула-подобных классов) так и с точки зрения времени вычислений (благодаря быстродействию Си). В начале в Си были добавлены классы (с инкапсуляцией), производные классы, строгая проверка типов, inline-функции и аргументы по умолчанию. Разрабатывая *Си с классами*  (позднее Си++), Страуструп также написал программу cfront — транслятор, перерабатывающий исходный код *Си с классами* в исходный код простого Си. Новый язык, неожиданно для автора, приобрёл большую популярность среди коллег и вскоре Страуструп уже не мог лично поддерживать его, отвечая на тысячи вопросов. В 1983 году произошло переименование языка из *Си с классами* в Си++. Кроме того, в него были добавлены новые возможности, такие как виртуальные функции, перегрузка функций и операторов, ссылки, константы, пользовательский контроль над управлением свободной памятью, улучшенная проверка типов и новый стиль комментариев (//). Его первый коммерческий выпуск состоялся в октябре 1985 года. В 1985 году вышло также первое издание «Языка программирования Си++», обеспечивающее первое описание этого языка, что было чрезвычайно важно из-за отсутствия официального стандарта. В 1989 году состоялся выход Си++ версии 2.0. Его новые возможности включали множественное наследование, абстрактные классы, статические функции-члены, функции-константы и защищённые члены.

В 1990 году вышло «Комментированное справочное руководство по C++», положенное впоследствии в основу стандарта. Последние обновления включали шаблоны, исключения, пространства имён, новые способы приведения типов и булевский тип.

Стандартная библиотека Си++ также развивалась вместе с ним. Первым добавлением к стандартной библиотеке Си++ стали потоки ввода/вывода, обеспечивающие средства для замены традиционных функций Си printf и scanf. Позднее самым значительным развитием стандартной библиотеки стало включение в неё Стандартной библиотеки шаблонов.

После многих лет работы совместный комитет ANSI-ISO стандартизировал Си++ в 1998 году (ISO/IEC 14882:1998 — Язык программирования Си++). В течение нескольких лет после официального выхода стандарта комитет обрабатывал сообщения об ошибках и в итоге выпустил исправленную версию стандарта Си++ в 2003 году. В настоящее время рабочая группа МОС (ISO) работает над новой версией стандарта под кодовым названием C++09 (ранее известный как C++0X), который должен выйти в 2009 году.

Никто не обладает правами на язык Си++, он является свободным. Однако сам документ стандарта языка (за исключением черновиков) не доступен бесплатно. Название «Си++» было придумано Риком Масситти (Rick Mascitti) и впервые было использовано в декабре 1983 года. Ранее, на этапе разработки, новый язык назывался «Си с классами».

Имя, получившееся в итоге, происходит от оператора Си «++» (увеличение значения переменной на единицу). Имя «C+» не было использовано потому, что является синтаксической ошибкой в Си и, кроме того, это имя было занято другим языком. Язык также не назван «D», поскольку «является расширением Си и не пытается устранять проблемы путём удаления элементов Си». Стандарт Си++ на 1998 год состоит из двух основных частей: ядра языка и стандартной библиотеки. Стандартная библиотека Си++ вобрала в себя разрабатывавшуюся одновременно со стандартом библиотеку шаблонов STL. Сейчас название STL официально не употребляется, однако в кругах программистов на Си++ это название используется для обозначения части стандартной библиотеки, содержащей определения шаблонов контейнеров, итераторов, алгоритмов и функторов. Стандарт Си++ содержит нормативную ссылку на стандарт Си от 1990 года и не определяет самостоятельно те функции стандартной библиотеки, которые заимствуются из стандартной библиотеки Си.

Кроме того, существует огромное количество библиотек Си++, не входящих в стандарт. В программах на Си++ можно использовать многие библиотеки Си.

Стандартизация определила язык программирования Си++, однако за этим названием могут скрываться также неполные, ограниченные, достандартные варианты языка. В первое время язык развивался вне формальных рамок, спонтанно, по мере ставившихся перед ним задач. Развитию языка сопутствовало развитие кросс-компилятора cfront. Новшества в языке отражались в изменении номера версии кросс-компилятора. Эти номера версий кросс-компилятора распространялись и на сам язык, но применительно к настоящему времени речь о версиях языка Си++ не ведут.

**JAVA**

Java — объектно-ориентированный язык программирования, разрабатываемый компаниейSun Microsystems. Приложения Java обычно компилируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине (JVM) независимо откомпьютерной архитектуры. Дата официального выпуска — 23 мая 1995 года.

В отношении произношения в русском языке, как и в ряде других, образовались две различные нормы — заимствованная англоязычная/ˈdʒɑːvə/(«джава») и традиционно-национальная «ява», соответствующая традиционному произношению названия острова Ява. Компания Sun придерживается англоязычного прозношения во всех странах мира. Иногда в обиходе используют также жаргонное слово «Жаба» (например, изображение жабы есть на календариках группы российский пользователей Java (Java Users Group).

Java — так называют не только сам язык, но и платформу для создания и исполнения приложений на основе данного языка.

Изначально язык назывался Oak («дуб») и разрабатывался Джеймсом Гослингом для программирования бытовых электронных устройств. Впоследствии он был переименован в Java и стал использоваться для написания клиентских приложений и серверного программного обеспечения. Назван в честь марки кофе Java, любимого некоторыми программистами, поэтому на официальной эмблеме языка изображена чашка с парящим кофе.

Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор, но с тем отличием, что байтовый код, в отличие от текста, обрабатывается значительно быстрее.

Достоинство подобного способа выполнения программ — в полной независимости байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности благодаря тому, что исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером) вызывают немедленное прерывание.

Часто к недостаткам концепции виртуальной машины относят то, что исполнение байт-кода виртуальной машиной может снижать производительность программ и алгоритмов, реализованных на языке Java. Данное утверждение было справедливо для первых версий виртуальной машины Java, однако в последнее время оно практически потеряло актуальность. Этому способствовал ряд усовершенствований:

* применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы (JIT-технология) с возможностью сохранения версий класса в машинном коде,
* широкое использование платформенно-ориентированного кода (native-код) в стандартных библиотеках,
* аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология Jazelle, поддерживаемая некоторыми процессорами фирмы ARM).

Идеи, заложенные в концепцию и различные реализации среды виртуальной машины Java, вдохновили множество энтузиастов на расширение перечня языков, которые могли бы быть использованы для создания программ, исполняемых на виртуальной машине.[13] Эти идеи нашли также выражение в спецификация общеязыковой инфраструктуры CLI, заложенной в основу платформы .NETкомпанией Microsoft.

**Заключение**

Каждый 256-й день года отмечается неофициальный праздник – День программиста. Это число (два в восьмой степени) выбрано неслучайно, так как это количество чисел, которые можно выразить с помощью одного байта. Сегодня программист – одна из самых востребованных профессий, ведь значение деятельности этих специалистов невозможно переоценить: ежедневно миллионы людей сталкиваются с результатами их труда, когда работают за компьютером, пользуются телефоном и смотрят телевизор.

Поэтому очень важно изучать то, с чем непосредственно эти люди работают – с языками программирования. Самые популярные и известные из них я описал в своем реферате.

**Литература**

1. Б. Страуструп. Язык программирования C++ = The C++ Programming Language / Пер. с англ. — 3-е изд. — СПб.; М.: Невский диалект — Бином, 1999. — 991 с. — 3000 экз. — ISBN 5-7940-0031-7 (Невский диалект), ISBN 5-7989-0127-0 (Бином), ISBN 0-201-88954-4 (англ.)
2. Страуструп Б. Язык программирования C++. Специальное издание = The C++ programming language. Special edition. — М.: Бином-Пресс, 2007. — 1104 с. — ISBN 5-7989-0223-4
3. Кей С. Хорстманн, Гари Корнелл. Java 2. Библиотека профессионала, том 1. Основы = Core Java 2, Volume I — Fundamentals. — 8-е изд. — М.: Вильямс, 2008. — 816 с. — ISBN 978-5-8459-1378-4, ISBN 978-0-1323-5476-9
4. Кей С. Хорстманн, Гари Корнелл. Java 2. Библиотека профессионала, том 2. Тонкости программирования = Core Java 2, Volume II — Advanced Features. — 8-е изд. — М.: Вильямс, 2008, 4 кв. — 992 с. — ISBN 978-5-8459-1482-8, ISBN 978-01-3235479-0
5. Монахов Вадим. Язык программирования Java и среда NetBeans. — 2-е изд. — СПб.: БХВ-Петербург, 2009. — 720 с. — ISBN 978-5-9775-0424-9
6. Роберт У. Себеста. Основные концепции языков программирования = Concepts of Programming Languages / Пер. с англ. — 5-е изд. — М.: Вильямс, 2001. — 672 с. — 5000 экз. — ISBN 5-8459-0192-8 (рус.), ISBN 0-201-75295-6 (англ.)
7. Вольфенгаген В. Э. Конструкции языков программирования. Приёмы описания. — М.: Центр ЮрИнфо Р, 2001. — 276 с. — ISBN 5-89158-079-9