МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
Высшего образования
«Хакасский государственный университет им. Н.Ф. Катанова»
Колледж педагогического образования, информатики и права

ПЦК естественнонаучных дисциплин, математики и информатики

РЕФЕРАТ

на тему:
Языки программирования

Автор реферата: _____ В.А.Винидиктова
(подпись) (инициалы, фамилия)

Специальность: 090203 - Программирование в компьютерных системах

Курс: II
Группа: И-21
Зачет/незачет:_____
Руководитель: _____ О.П.Когумбаева
(подпись, дата) (инициалы, фамилия)

г. Абакан, 2018г.

Содержание:

Введение

Язык программирования (ЯП) - формальный язык, предназначенный для записи компьютерных программ; определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель под её управлением.

ЯП предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной вычислительный процесс, организовать управление различными объектами. Такой язык отличается от естественных языков тем, что предназначен для управления ЭВМ, в то время как естественные языки используются, прежде всего, для общения людей между собой.

В ходе данного исследования были проанализированы языки программирования: определение, основная их классификация, история появления ЯП, и примеры известных сейчас ЯП.

Целью реферата является исследование характеристик и особенностей различных ЯП.

Для достижения цели необходимо выполнить ряд следующих задач:

1. Рассмотреть определение «Язык программирования» и основные понятия данной области;
2. Исследовать историю создания первых ЯП;
3. Рассмотреть классификацию ЯП;
4. Исследовать наиболее популярные ЯП и их характеристики.

1. Язык программирования

1.1. Определение

Язык программирования - формальный язык, предназначенный для записи компьютерных программ. ЯП определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель. ЯП предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной вычислительный процесс, организовать управление различными объектами, и т. п. ЯП отличается от естественных языков тем, что предназначен для управления ЭВМ, в то время как естественные языки используются, прежде всего, для общения людей между собой.

1.2. История создания

Можно сказать, что первые ЯП возникали ещё до появления современных электронных вычислительных машин: уже в XIX веке были изобретены устройства, которые можно с долей условности назвать программируемыми — к примеру, музыкальная шкатулка посредством металлического цилиндра и Жаккардовый ткацкий станок (1804) посредством картонных карт. Для управления ими использовались наборы инструкций, которые в рамках современной классификации можно считать прототипами предметно-ориентированных языков программирования. Значимым можно считать «язык», на котором леди Ада Лавлейс в 1842 году написала программу для вычисления чисел Бернулли для Аналитической машины Чарльза Бэббиджа, ставшей бы, в случае реализации, первым компьютером в мире.

В 1930—1940 годах, А. Чёрч, А. Тьюринг, А. Марков разработали математические абстракции — для формализации алгоритмов.

В это же время, в 1940-е годы, появились электрические цифровые компьютеры и был разработан язык, который можно считать первым высокоуровневым языком программирования для ЭВМ — «Plankalkül», созданный немецким инженером К. Цузе в период с 1943 по 1945 годы.

Первым практически реализованным языком стал в 1949 году так называемый «Краткий код», в котором операции и переменные кодировались двухсимвольными сочетаниями. Он был разработан в компании Eckert–Mauchly Computer Corporation, выпускавшей UNIVAC-и, созданной одним из сотрудников Тьюринга, Джоном Мокли.

Вскоре на смену такому методу программирования пришло применение языков второго поколения, также ограниченных спецификациями конкретных машин, но более простых для использования человеком за счёт

использования мнемоник (символьных обозначений машинных команд) и возможности сопоставления имён адресам в машинной памяти. Они традиционно известны под наименованием языков ассемблера и автокодов. Однако, при использовании ассемблера становился необходимым процесс перевода программы на язык машинных кодов перед её выполнением, для чего были разработаны специальные программы, также получившие название ассемблеров. Сохранялись и проблемы с переносимостью программы с ЭВМ одной архитектуры на другую, и необходимость для программиста при решении задачи мыслить терминами «низкого уровня» — ячейка, адрес, команда. Позднее языки второго поколения были усовершенствованы: в них появилась поддержка макрокоманд.

С середины 1950-х начали появляться языки третьего поколения, такие как Фортран, Лисп и Кобол. ЯП этого типа более абстрактны (их ещё называют «языками высокого уровня») и универсальны, не имеют жёсткой зависимости от конкретной аппаратной платформы и используемых на ней машинных команд. Программа на языке высокого уровня может исполняться на любой ЭВМ, на которой для этого языка имеется транслятор (инструмент, переводящий программу на язык машины, после чего она может быть выполнена процессором).

Обновлённые версии перечисленных языков до сих пор имеют хождение в разработке программного обеспечения, и каждый из них оказал определенное влияние на последующее развитие языков программирования. Тогда же, в конце 1950-х годов, появился Алгол, также послуживший основой для ряда дальнейших разработок в этой сфере. Необходимо заметить, что на формат и применение ранних языков программирования в значительной степени влияли интерфейсные ограничения.

2. Классификация ЯП

Не существует общепринятой систематичной таксономии языков программирования. Есть множество черт, согласно которым можно производить классификацию языков, причём одни из них однозначно проводят разделы между языками на основе технических свойств; другие основываются на доминирующих признаках, имеют исключения и более условны; а третьи полностью субъективны и нередко сопровождаются заблуждениями, но на практике весьма распространены.

2.1. Языки программирования высокого уровня

Высокоуровневый язык программирования — ЯП, разработанный для быстроты и удобства использования программистом. Основная черта высокоуровневых языков — это абстракция, то есть введение смысловых конструкций, кратко описывающих такие структуры данных и операции над ними, описания которых на машинном коде (или другом низкоуровневом языке программирования) очень длинны и сложны для понимания.

Так, высокоуровневые языки стремятся не только облегчить решение сложных программных задач, но и упростить портирование программного обеспечения. Использование разнообразных трансляторов и интерпретаторов обеспечивает связь программ, написанных при помощи языков высокого уровня, с различными операционными системами и оборудованием, в то время как их исходный код остаётся, в идеале, неизменным.

Примеры высоких языков:

- C++
- C#
- Visual Basic
- Python

- Perl
- Fortran
- Delphi(Pascal)
- Lisp

2.2. Языки программирования низкого уровня

Низкоуровневый язык программирования — ЯП, близкий к программированию непосредственно в машинных кодах используемого реального или виртуального (например, Java, Microsoft .NET) процессора. Для обозначения машинных команд обычно применяется мнемоническое обозначение. Это позволяет запоминать команды не в виде последовательности двоичных нулей и единиц, а в виде осмысленных сокращений слов человеческого языка (обычно английских). Как правило, использует особенности конкретного семейства процессоров. Общеизвестный пример низкоуровнего языка — язык ассемблера, хотя правильнее говорить о группе языков ассемблера. Для одного и того же процессора существует несколько видов языка ассемблера. Они совпадают в машинных командах, но различаются набором дополнительных функций (директив и макросов).

2.3. Компилируемые и интерпретируемые языки

Программа на компилируемом языке при помощи специальной программы компилятора преобразуется в набор инструкций для данного типа процессора 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, 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.

Примеры компилированных языков: assembler, C++, Pascal

Примеры интерпритируемых языков: PHP, JavaScript, Python

Некоторые языки, например, Java и C#, находятся между компилируемыми и интерпретируемыми.

2.4. Безопасные и небезопасные языки

В общем и целом, язык называется безопасным, если программы на нём, которые могут быть приняты компилятором как правильно построенные, в динамике никогда не выйдут за рамки допустимого поведения. Это не значит, что такие программы не содержат ошибок вообще. Термин «хорошее поведение программы» (англ. well behavior) означает, что даже если программа содержит некий баг (в частности, логическую ошибку), она не способна нарушить целостность данных и обрушиться. Хотя термины неформальны, безопасность некоторых языков (например, Standard ML) математически доказуема. Безопасность других (например, Ada) была обеспечена ad hoc-образом, без обеспечения концептуальной целостности, что может обернуться катастрофами, если положиться на них в ответственных задачах .

Языки C и его потомок C++ являются небезопасными. В программах на них обширно встречаются ситуации ослабления типизации (приведение типов) и прямого её нарушения (каламбур типизации), так что ошибки доступа к памяти являются в них статистической нормой. Самые мощные системы статического анализа для них способны обнаруживать не более 70 — 80 % ошибок, но их использование обходится очень дорого в денежном смысле. Достоверно же гарантировать безотказность программ на этих языках невозможно, не прибегая к формальной верификации, что не только ещё дороже, но и требует специальных знаний.

У Си есть и безопасные потомки, такие как Cyclone или Rust. Язык Forth не претендует на звание «безопасного», но, тем не менее, на практике существование программ, способных повредить данные, почти исключено, так как содержащая потенциально опасную ошибку программа аварийно завершается на первом же тестовом запуске, принуждая к коррекции исходного кода. В сообществе Erlang принят подход «let it crash» , также нацеленный на раннее выявление ошибок.

3. Примеры современных языков программирования

3.1 Си его разновидности

Си [C] - Многоцелевой язык программирования высокого уровня, разработанный Денисом Ритчи в начале 1970-х гг. на базе языка BCPL. Используется на миниЭВМ 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) - 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.

3.2. JavaScript

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания — что придаёт языку дополнительную гибкость.

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком Си имеет коренные отличия:

- объекты с возможностью интроспекции;
- функции как объекты первого класса;
- автоматическое приведение типов;
- автоматическая сборка мусора;
- анонимные функции.

В языке отсутствуют такие полезные вещи, как:

- модульная система: JavaScript не предоставляет возможности управлять зависимостями и изоляцией областей видимости;
- стандартная библиотека: в частности, отсутствует интерфейс программирования приложений по работе с файловой системой, управлению потоками ввода-вывода, базовых типов для бинарных данных;
- стандартные интерфейсы к веб-серверам и базам данных;
- система управления пакетами, которая бы отслеживала зависимости и автоматически устанавливала их.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

3.3. Python

Python (в русском языке распространено название питон) - высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python — активно развивающийся язык программирования, новые версии (с добавлением/изменением языковых свойств) выходят примерно раз в два с половиной года. Вследствие этого и некоторых других причин на Python отсутствуют стандарт ANSI, ISO или другие официальные стандарты, их роль выполняет CPython.

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. Есть реализации интерпретаторов для JVM (с возможностью компиляции), MSIL (с возможностью компиляции), LLVM и других.

Заключение

Целью реферата было исследование языков программирования и их характеристик. Был поставлен ряд задач, которые необходимо было выполнить для достижения намеченной цели. Если рассмотреть последовательно каждый пункт, то можно сделать вывод, что цель реферата достигнута: были исследованы характеристики языков программирования. Даны ответы на вопросы: что такое язык программирования; история создания; классификация; так же были рассмотрены некоторые современные языки программирования.

В настоящее время языки программирования активно участвуют в жизни человека. С их развитием намного облегчается работа не только программистов, но и обычных людей. На основании всего этого можно сделать вывод, что в данное время без языков программирования не обойтись.

Библиографический список

1. Керниган Брайан, Донован Алан: Язык программирования Go. Изд-во: Вильямс, 2016 г. - 432 с.

2. Страуструп Бьерн: Язык программирования C++. Изд-во: Бином, 2017 г. - 1136 с.

3. Питер Принц, Кроуфорд Тони : Язык C. Справочник. Полное описание языка. Изд-во Вильямс, 2017 г. - 880 с.

4. Кабаков Роберт: R в действии. Анализ и визуализация данных на языке R. Изд-во: ДМК-Пресс, 2016 г. - 588 с.

5. Стил Гай, Гослинг Джеймс, Джой Билл: Язык программирования Java SE 8. Подробное описание. Изд-во: Вильямс, 2015 г. - 672 с.

6. Керниган Брайан, Ритчи Деннис: Язык программирования C. Изд-во: Вильямс, 2016 г. - 288 с.

7. Ян Солем: Программирование компьютерного зрения на языке Python. Изд-во: ДМК-Пресс, 2016 г. - 312 с.

8. Фаулер Мартин:Предметно-ориентированные языки программирования. Изд-во: Вильямс, 2013 г. - 576 с.

9. Clojure (язык программирования) // Материал из национальной библиотеки Н.Э. Баумана. Последнее изменение этой страницы: 23:18, 8 июня 2016. [Электронный ресурс]. URL: https://ru.bmstu.wiki/Clojure_(язык_программирования). (Дата обращения 17.01.2018).

10. GLSL (OpenGL Shading Language) // Материал из национальной библиотеки Н.Э. Баумана. Последнее изменение этой страницы: 02:25, 22 мая 2016. [Электронный ресурс]. URL: https://ru.bmstu.wiki/GLSL_(OpenGL_Shading_Language).(Дата обращения 17.01.2018).