1. Io 2

1.1. Общая характеристика и история создания 2

1.2. Система типов и особенности архитектуры языка 2

1.3. Примеры кода и интересные факты 2

5. Lolcode 2

5.1. Общая характеристика и история создания 2

5.2. Система типов и особенности архитектуры языка 3

5.3. Примеры кода и интересные факты 3

18. Dart 3

18.1. Общая характеристика и история создания 3

18.2. Система типов и особенности архитектуры языка 4

18.3. Примеры кода и интересные факты 4

4. Список литературы 6

# Io

## Общая характеристика и история создания

«Io (первая буква — заглавная «i») — скриптовый, чисто объектно-ориентированный язык программирования с динамической типизацией. Язык был разработан Стивом Декортом (англ. Steve Dekorte) в 2002 году. Реализация является кроссплатформенной, открытой (лицензия BSD) и рассчитана на лёгкость встраивания в качестве скриптового языка.»[1]

## Система типов и особенности архитектуры языка

«Особенности

* минималистический синтаксис;
* абсолютно каждая сущность является объектом;
* всё основано на передаче сообщений (даже арифметические операции и присваивание — это передача сообщения объекту);
* параллелизм, основанный на модели акторов.

В отличие от Perl, PHP, Python и Ruby, в Io класс не имеет принципиального отличия от объекта, является первоклассным объектом языка. Наследование реализовано прототипами, как в Self, Lua и JavaScript. Поддерживается множественное наследование.

Таким образом, для экземпляров объектов, классов и пространств имён в Io используются прототипы. Для операторов, вызовов методов, присваивания и доступа к переменным используются сообщения.»[2]

## Примеры кода и интересные факты

«Следующий код печатает в консоли «Hello world!» посредством передачи строке сообщения println:

1. "Hello world!" println

Реализация шаблона проектирования «одиночка»:

1. Singleton := Object **clone** *//Объекты создаются (":=") клонированием существующих объектов*
2. Singleton **clone** = Singleton *//Значение слота clone заменяется ("=") ссылкой на сам объект»*[3]

# Lolcode

## Общая характеристика и история создания



Рисунок 1

«LOLCODE — эзотерический язык программирования, созданный под влиянием интернет-мема о lolcat’ах (см на Рисунок 1). На официальном сайте находится спецификация этого языка, и, хотя некоторые моменты в ней не описаны, уже существуют несколько работающих интерпретаторов и компиляторов этого языка. LOLCODE является Тьюринг-полным языком, так как возможна реализация интерпретатора Brainfuck на LOLCODE.»[3]

## Система типов и особенности архитектуры языка

«Программа заключена в ключевые слова HAI и KTHXBYE. После слова HAI может быть указана версия спецификации языка, которая используется в программе, но не указано, как интерпретатор должен ее обрабатывать.

Комментарии используют слова BTW (комментарий до конца строки) и OBTW ... TLDR (комментарий, который может занимать несколько строк).

I HAS A <varname> — объявление переменной. LOLCODE динамически типизирован, поэтому нет необходимости объявлять еще и тип переменной.

<varname> R <value> — присвоение переменной значения. В языке есть следующие типы данных: NOOB (нетипизированный — это значение принимает переменная до того, как она инициализирована), NUMBR (целое), NUMBAR (дробное), YARN (строка) и TROOF (логический). Тип BUKKIT зарезервирован на будущее для массивов. Допускается автоматическое преобразование типов.

Для функций и операторов используется префиксная нотация. (см. Таблица 1) Если количество параметров функции может варьироваться, их список должен оканчиваться ключевым словом MKAY. Отдельные аргументы могут разделяться словом AN.»[5]

## Примеры кода и интересные факты

«LOLCODE использует сильно урезанный словарь английского интернет-сленга, поэтому человек, знакомый с этим сленгом, вполне может разобраться в коде программы даже без знания синтаксиса. Вот пример программы «Hello world» и простой программы по выводу содержимого файла на экран:

1. HAI
2. CAN HAS STDIO?
3. VISIBLE "HAI WORLD!"
4. KTHXBYE
5. HAI
6. CAN HAS STDIO?
7. PLZ OPEN FILE "LOLCATS.TXT"?
8. AWSUM THX
9. VISIBLE FILE
10. O NOES
11. VISIBLE "ERROR!"
12. KTHXBYE

»[6]

|  |  |
| --- | --- |
| **Комментарий до конца строки** | BTW |
| **Комментарии, которые не могут быть вложенными** | OBTW ... TLDR |
| **Регистрозависимость** | да |
| **Регулярное выражение идентификатора переменной** | [a-zA-Z]\* |
| **Присваивание значения переменной** | <varname> R <value> |
| **Объявление переменной** | I HAS A <varname> |
| **Равенство** | BOTH SAEM |
| **Неравенство** | DIFFRINT |
| **Если - то** | <expression> O RLY? YA RLY <code block> OIC |
| **Если - то - иначе** | <expression> O RLY? YA RLY <code block> NO WAI <code block> OIC |
| **Бесконечный цикл** | IM IN YR <label> ... IM OUTTA YR <label> |

Таблица 1

# Dart

## Общая характеристика и история создания



Рисунок 2

«Dart — язык программирования, созданный Google. Dart позиционируется в качестве замены/альтернативы JavaScript. Один из разработчиков языка Марк Миллер (Mark S. Miller) написал, что JavaScript «имеет фундаментальные изъяны» («Javascript has fundamental flaws…»), которые невозможно исправить. Поэтому и был создан Dart. Первая общедоступная информация об этом языке программирования появилась 12 сентября 2011 года на конференции разработчиков Goto[2]. 10 октября 2011 была проведена официальная презентация языка Google Dart. Изначально было предложено два способа исполнения Dart-программ: с использованием виртуальной машины (в поддерживающих язык браузерах) или с промежуточной трансляцией в javascript (универсальный)[3][4]. 15 ноября 2013 года Google выпустили первую стабильную версию своего языка программирования — Dart SDK 1.0 . 4 июля 2014 года ECMA одобрили первую редакцию стандарта языка, стандарт получил название ECMA-408.»[7]

## Система типов и особенности архитектуры языка

«Сейчас Dart (см. Рисунок 2) находится в самом начале своего развития. Следующие цели будут направлять язык в его развитии:  
— Создать структурированный и в тоже время гибкий язык программирования для веб  
— Сделать Dart знакомым и естественным для программистов  
— Сделать так, чтобы все конструкции языка обеспечивали высокую производительность и быстрый старт приложения  
— Сделать возможным запускать Dart на всевозможных веб-устройствах, включая телефоны, планшеты, компьютеры и серверы.  
— Предоставить инструменты, которые позволят Dart выполняться на всех основных современных браузерах.  
  
Эти цели решают следующие проблемы, с которыми часто встречаются веб-разработчики:  
— Маленькие скрипы очень часто мутируют в огромные приложения без какой-либо структуры их очень сложно отлаживать и проблематично поддерживать. В дополнении, эти монолитные приложения не могут быть разделены на адекватные части, поэтому разные команды разработчиков не могут работать над ними раздельно. Веб-приложения невозможно продуктивно разрабатывать, когда они становятся слишком большими.  
— Скриптовые языки популярны потому, что их легкая структура позволяет быстро писать код. Как правило, договоренности между частями приложения передаются в виде комментариев, вместо того, чтобы использовать языковые структуры. В результате, другим разработчикам сложно понять, читать и поддерживать такой код (см. Таблица 2).  
— Существующие языки делят разработчиков на 2 лагеря: приверженцев статических и динамических языков. Традиционные статические языки требуют тяжелых инструментов и сложных стилей кодирования, которые ограничивают разработчика.  
— Разработчики не могут создавать однородные системы, которые охватывают как клиент и сервер, за исключением нескольких классов для Node.js и Google Web Toolkit (GWT).  
— Разные языки и форматы, влекут за собой переключение контекста, что усложняет процесс кодирования.»[8]

| **Library** | **Notes** |
| --- | --- |
| [dart:async](https://api.dart.dev/stable/dart-async/dart-async-library.html) Support for asynchronous programming, with classes such as Future and Stream. |  |
| [dart:collection](https://api.dart.dev/stable/dart-collection/dart-collection-library.html) Classes and utilities that supplement the collection support in dart:core. |  |
| [dart:convert](https://api.dart.dev/stable/dart-convert/dart-convert-library.html) Encoders and decoders for converting between different data representations, including JSON and UTF-8. |  |
| [dart:core](https://api.dart.dev/stable/dart-core/dart-core-library.html) Built-in types, collections, and other core functionality for every Dart program. |  |
| [dart:developer](https://api.dart.dev/stable/dart-developer/dart-developer-library.html) Interaction with developer tools such as the debugger and inspector. | JIT and dartdevc only |
| [dart:math](https://api.dart.dev/stable/dart-math/dart-math-library.html) Mathematical constants and functions, plus a random number generator. |  |
| [dart:typed\_data](https://api.dart.dev/stable/dart-typed_data/dart-typed_data-library.html) Lists that efficiently handle fixed sized data (for example, unsigned 8-byte integers) and SIMD numeric types. |  |

Таблица 2

## Примеры кода и интересные факты

«Простое приложение, выводящее надпись Hello, world!:

1. main() {
2. print('Hello World!');
3. }

Функция для расчета n-го числа Фибоначчи:

1. int fib(int n) {
2. **return** (n <= 2) ? 1 : (fib(n - 1) + fib(n - 2));
3. }
4. main() {
5. print('fib(20) = ${fib(20)}');
6. }

Простой класс:

1. *// Подключить библиотеку math для доступа к функции sqrt()*
2. **import** 'dart:math' as math;
3. *// Создание класса Point*
4. **class** Point {
5. *// Переменные, объявленные как final, не могут быть изменены после инициализации*
6. *// Объявление двух числовых полей*
7. **final** num x, y;
8. *// Конструктор c "синтаксическим сахаром", инициализирующий поля объекта значениями аргументов*
9. Point(**this**.x, **this**.y);
10. *// Именованный конструктор со списком инициализации по умолчанию*
11. Point.origin() : x = 0, y = 0;
12. *// Метод класса*
13. num distanceTo(Point other) {
14. **var** dx = x - other.x;
15. **var** dy = y - other.y;
16. **return** math.sqrt(dx \* dx + dy \* dy);
17. }
18. }
19. *// Все программы на Dart начинают выполнение с функции main()*
20. main() {
21. *// Создание двух объектов класса Point*
22. **var** p1 = **new** Point(10, 10);
23. **var** p2 = **new** Point.origin();
24. **var** distance = p1.distanceTo(p2);
25. print(distance);
26. }»[9]

# Список литературы

1. <https://ru.wikipedia.org/wiki/Io>
2. <https://ru.wikipedia.org/wiki/Io#Особенности>
3. <https://ru.wikipedia.org/wiki/Io#Примеры>
4. <https://ru.wikipedia.org/wiki/LOLCODE>
5. <https://ru.wikipedia.org/wiki/LOLCODE#Особенности>
6. <https://ru.wikipedia.org/wiki/LOLCODE#Примеры>
7. <https://ru.wikipedia.org/wiki/Dart>
8. <https://habr.com/ru/post/130064/>
9. <https://ru.wikipedia.org/wiki/Dart#Примеры>