

<https://habr.com/ru/post/223253/> **domu** 18 мая 2014 в 22:23

«Забывшие» парадигмы программирования

Разработка веб-сайтов, Программирование, Совершенный код



Получилось так, что те парадигмы, которые раньше потом и кровью пробивались в свет через орды приверженцев традиционных методов постепенно забываются. Эти парадигмы возникли на заре программирования и то, почему они возникали, какие преимущества они давали и почему используются до сих пор полезно знать любому разработчику.

Ладно. Введение это очень весело, но вы его все равно не читаете, так что кому интересно — добро пожаловать под кат!

Императивное программирование



Исторически сложилось так, что подавляющее большинство вычислительной техники, которую мы программируем имеет состояние и программируется инструкциями, поэтому первые языки программирования в основном были чисто императивными, т.е. не поддерживали никаких парадигм кроме императивной.

Это были машинные коды, языки ассемблера и ранние высокоуровневые языки, вроде Fortran.

Ключевые моменты:

В этой парадигме вычисления описываются в виде инструкций, шаг за шагом изменяющих состояние программы.

В низкоуровневых языках (таких как язык ассемблера) состоянием могут быть память, регистры и флаги, а инструкциями — те команды, что поддерживает целевой процессор.

В более высокоуровневых (таких как Си) состояние — это только память, инструкции могут быть сложнее и вызывать выделение и освобождение памяти в процессе своей работы.

В совсем высокоуровневых (таких как Python, если на нем программировать императивно) состояние ограничивается лишь

переменными, а команды могут представлять собой комплексные операции, которые на ассемблере занимали бы сотни строк.

Основные понятия:

- Инструкция
- Состояние

Порожденные понятия:

- Присваивание
- Переход
- Память
- Указатель

Языки поддерживающие данную парадигму:

Как основную:

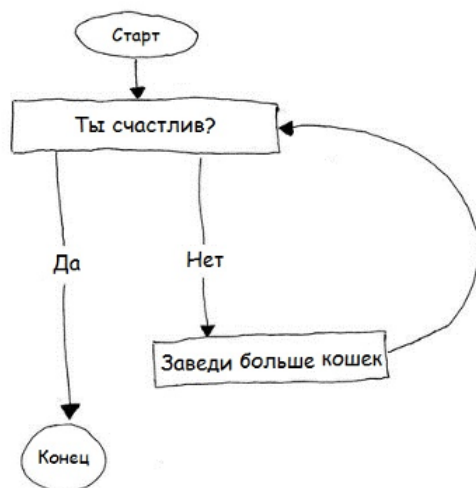
- Языки ассемблера
- Fortran
- Algol
- Cobol
- Pascal
- C
- C++
- Ada

Как вспомогательную:

- Python
- Ruby
- Java
- C#
- PHP
- Haskell (через монады)

Стоит заметить, что большая часть современных языков в той или иной степени поддерживает императивное программирование. Даже на чистом функциональном языке Haskell можно писать императивно.

Структурное программирование



Структурное программирование — парадигма программирования (также часто встречающееся определение — методология разработки), которая была первым большим шагом в развитии программирования.

Основоположниками структурного программирования были такие знаменитые люди как Э. Дейкстра и Н. Вирт.

Языками-первопроходцами в этой парадигме были Fortran, Algol и В, позже их приемниками стали Pascal и С.

Ключевые моменты:

Эта парадигма вводит новые понятия, объединяющие часто используемые шаблоны написания императивного кода.

В структурном программировании мы по-прежнему оперируем состоянием и инструкциями, однако вводится понятие составной инструкции (блока), инструкций ветвления и цикла.

Благодаря этим простым изменениям возможно отказаться от оператора `goto` в большинстве случаев, что упрощает код.

Иногда `goto` все-же делает код читабельнее, благодаря чему он до сих пор широко используется, несмотря на все заявления его противников.

Основные понятия:

- Блок
- Цикл
- Ветвление

Языки поддерживающие данную парадигму:

Как основную:

- С
- Pascal
- Basic

Как вспомогательную:

- С#
- Java
- Python
- Ruby
- JavaScript

Поддерживают частично:

- Некоторые макроассемблеры (через макросы)

Опять-же большая часть современных языков поддерживают структурную парадигму.

Процедурное программирование



Опять-же возрастающая сложность программного обеспечения заставила программистов искать другие способы описывать вычисления.

Собственно еще раз были введены дополнительные понятия, которые позволили по-новому взглянуть на программирование.

Этим понятием на этот раз была процедура.

В результате возникла новая методология написания программ, которая приветствуется и по сей день — исходная задача разбивается на меньшие (с помощью процедур) и это происходит до тех пор, пока решение всех конкретных процедур не окажется тривиальным.

Ключевые моменты:

Процедура — самостоятельный участок кода, который можно выполнить как одну инструкцию.

В современном программировании процедура может иметь несколько точек выхода (return в C-подобных языках), несколько точек входа (с помощью yield в Python или статических локальных переменных в C++), иметь аргументы, возвращать значение как результат своего выполнения, быть перегруженной по количеству или типу параметров и много чего еще.

Основные понятия:

— Процедура

Порожденные понятия:

— Вызов
— Аргументы
— Возврат
— Рекурсия
— Перегрузка

Языки поддерживающие данную парадигму:

Как основную:

— C
— C++
— Pascal
— Object Pascal

Как вспомогательную:

— C#
— Java
— Ruby
— Python
— JavaScript

Поддерживают частично:

— Ранний Basic

Стоит отметить, что несколько точек входа из всех этих языков поддерживаются только в Python.

Модульное программирование



Который раз увеличивающаяся сложность программ заставила разработчиков разделять свой код. На этот раз процедур было недостаточно и в этот раз было введено новое понятие — модуль.

Забегая вперед скажу, что модули тоже оказались неспособны сдержать с невероятной скоростью растущую сложность ПО и в последствии появились пакеты (это тоже модульное программирование), классы (это уже ООП), шаблоны (обобщенное программирование).

Программа описанная в стиле модульного программирования — это набор модулей. Что внутри, классы, императивный код или чистые функции — не важно.

Благодаря модулям впервые в программировании появилась серьезная инкапсуляция — возможно использовать какие-либо сущности внутри модуля, но не показывать их внешнему миру.

Ключевые моменты:

Модуль — это отдельная именованная сущность программы, которая объединяет в себе другие программные единицы, близкие по функциональности.

Например файл `List.mod` включающий в себя класс `List` и функции для работы с ним — модуль.

Папка `Geometry`, содержащая модули `Shape`, `Rectangle` и `Triangle` — тоже модуль, хоть и некоторые языки разделяют понятие модуля и пакета (в таких языках пакет — набор модулей и/или набор других пакетов).

Модули можно импортировать (подключать), для того, чтобы использовать объявленные в них сущности.

Основные понятия:

- Модуль
- Импортирование

Порожденные понятия:

- Пакет
- Инкапсуляция

Языки поддерживающие данную парадигму:

Как основную:

- Haskell
- Pascal
- Python

Как вспомогательную:

— Java
— C#
— ActionScript 3

Поддерживают частично:

— C/C++

В некоторых языках для модулей введены отдельные абстракции, в других же для реализации модулей можно использовать заголовочные файлы (в C/C++), пространства имен, статические классы и/или динамически подключаемые библиотеки.

Вместо заключения

В данной статье я не описал популярные сейчас объектно-ориентированное, обобщенное и функциональное программирование. Просто потому, что у меня есть свое, довольно радикальное мнение на этот счет и я не хотел разводить холивар. По крайней мере сейчас. Если тема окажется полезной для сообщества я планирую написать несколько статей, изложив основы каждой из этих парадигм подробно.

Также я ничего не написал про экзотические парадигмы, вроде автоматного, аппликативного, аспект/агент/компонент-ориентированного программирования. Я не хотел делать статью сильно большой и опять-же если тема будет востребована, я напишу и об этих парадигмах, возможно более подробно и с примерами кода.

Теги: парадигмы программирования, программирование, императивное программирование, структурное программирование, процедурное программирование

Хабы: Разработка веб-сайтов, Программирование, Совершенный код

↑ 0 ↓ 161 👁 64,6k 💬 18 ➦ Поделиться



53,0

Карма

0,0

Рейтинг

@domu

Пользователь

ПОХОЖИЕ ПУБЛИКАЦИИ

3 мая 2017 в 15:29

Шесть парадигм программирования, которые изменят ваш взгляд на код

↑ +32 👁 51,3k 📖 224 💬 49

20 января 2017 в 16:23

Один из простых способов улучшить свои навыки программирования — читать чужой код

↑ +36 👁 46,9k 📖 281 💬 34

7 декабря 2016 в 10:38

История языков программирования: Algol — жертва конфликта интересов

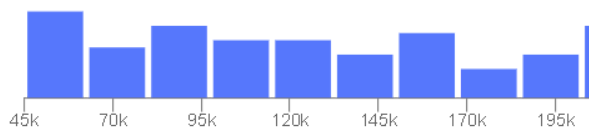
↑ +25 👁 14,4k 📖 34 💬 8

СРЕДНЯЯ ЗАРПЛАТА В IT



113 000 ₽/мес.

Средняя зарплата по всем IT-специализациям на основании 5 381 анкеты, за 2-ое пол. 2020 года

Узнать свою зарплату



Комментарии 18


 **workless** 18 мая 2014 в 23:34    +3 

Странно, C# в процедурном.


>>Стоит отметить, что несколько точек выхода из всех этих языков поддерживаются только в Python
что? оО

 **domu** 18 мая 2014 в 23:35     +2 


Упс, опечатка. Имелось ввиду несколько точек входа. И то с помощью yield.

 **lair** 19 мая 2014 в 00:48     +1 

Про существование yield (и заодно await) в C# вы не в курсе?

 **domu**  19 мая 2014 в 00:59     +2 

Теперь в курсе, спасибо, почитаю.

 **Fesor** 19 мая 2014 в 01:23     +6 

в php так же есть yield, и в javascript тоже (если брать ES6), да и в ruby...

 **AterCattus** 30 мая 2014 в 01:34     0 

Можно и lua до кучи :)

 **potan** 19 мая 2014 в 00:23    +4 



В императивной парадигме стоило бы упомянуть Fort.
В модульной — Ada и OCaml.

 **enkryptor** 30 января 2015 в 13:34     +1 

Forth*

 **potan** 30 января 2015 в 13:56     0 

Помню легенду, что 'h' было отброшено, что бы название в два слова поместилось. Но подтверждения найти не могу.

 **ApeCoder** 27 мая 2019 в 16:27     0 

Это и было отброшено

Forth is so named because in 1968 "the file holding the interpreter was labeled FOURTH, for 4th (next) generation software—but the IBM 1130 operating system restricted file names to 5 characters." [10]



sferrka 19 мая 2014 в 01:24

↑ +1 ↓

Процедура

возвращать значение как результат своего выполнения

Процедура != подпрограмма и не возвращает значения, просто выполняет список инструкций. Состояние же изменяется за счет глобальных переменных и передачи аргумента по ссылке.



sse 19 мая 2014 в 02:14

↑ +6 ↓

Классифицировано так грубо и условно, что в процессе совершенно исказился смысл.

К примеру, совершенно непонятно, как противопоставлять модульное программирование и ООП, если ООП — это разновидность модульного программирования, когда единственной единицей модульности является класс. Который — полноценная лингвистическая конструкция с одной стороны, и тип данных (в смысле абстрактных типов данных, по Вирту, т.е. тут уже и процедурное программирование проглядывает) — с другой.

Или другой некорректный пример — Си ++: в отличие от Си, тут файл (.cpp) вовсе не является модулем, а заголовок (.hpp) не является интерфейсом такого модуля. А в Си — являлся, и назывался объектом, единицей трансляции. В этом отношении (но не в других) Си ++ оказывается дальше от ООП, чем классический Си, что забавно.



knott 19 мая 2014 в 09:34

↑ 0 ↓

А нас, кстати, так учили.



potan 19 мая 2014 в 11:07

↑ +1 ↓

В TAPL разбираются отличия ООП и модулей — с точки зрения теории типов они оказываются двойственными.



Honeymann 19 мая 2014 в 15:36

↑ +3 ↓

К примеру, совершенно непонятно, как противопоставлять модульное программирование и ООП, если ООП — это разновидность модульного программирования, когда единственной единицей модульности является класс.

И вообще, все перечисленные парадигмы являются разновидностями императивной парадигмы. Помимо которой вполне себе заметны и развиты декларативная парадигма, парадигма метапрограммирования и событийно-ориентированная парадигма.

Вообще, достаточно хороший список есть на ru.wikipedia.org/wiki/Парадигма_программирования



gatoazul 9 октября 2014 в 22:02

↑ 0 ↓

Не ООП вообще разновидность модульного программирования, с классом=модулю, а только ООП в его общеизвестном виде. Это свойство распространенных сейчас языков, идущее, очевидно, из Явы, не более того.



kirillplatonov 19 мая 2014 в 08:02

↑ +1 ↓

В руби есть yield и модули



toxicdream 19 мая 2014 в 20:33

↑ +1 ↓

О как Pascal «универсален» — сразу во всех категориях!

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.