

Основы программирования

Занятие 2

TECT № 1

<http://goo.gl/forms/y6bS0o8oKV>

Классификация языков программирования

Классификация по парадигме

Парадигма программирования —
совокупность подходов к
написанию программного кода,
определяющая его стиль

Наиболее общая классификация

- Императивное программирование
- Декларативное программирование

Более подробная классификация

- Процедурные ЯП
- Функциональные ЯП
- Объектно-ориентированные ЯП
 - Основанные на классах (инкапсуляция, наследование, полиморфизм)
 - Основанные на прототипах (наследование через клонирование прототипа)
- Мультипарадигменные

Процедурные языки

Императивные

Основная идея: последовательное изменение состояние памяти

Компьютер = ЦПУ + память

Машина Тьюринга

Архитектура фон Неймана, гарвардская архитектура

Процедурные языки

- Ada
- Алгол 60
- Алгол 68
- Basic
- Си
- КОБОЛ
- Фортран
- Модула-2
- HAL/S
- Pascal
- PureBasic
- ПЛ/1
- Рапира
- REXX

КОБОЛ

```
$ SET SOURCEFORMAT "FREE"
```

```
IDENTIFICATION DIVISION.
```

```
PROGRAM-ID. Multiplier.
```

```
AUTHOR. Michael Coughlan.
```

```
* Example program using ACCEPT, DISPLAY and MULTIPLY to  
* get two single digit numbers from the user and multiply them  
* together
```

```
DATA DIVISION.
```

```
WORKING-STORAGE SECTION.
```

```
01 Num1 PIC 9 VALUE ZEROS.
```

```
01 Num2 PIC 9 VALUE ZEROS.
```

```
01 Result PIC 99 VALUE ZEROS.
```

```
PROCEDURE DIVISION.
```

```
    DISPLAY "Enter first number (1 digit) : " WITH NO ADVANCING.
```

```
    ACCEPT Num1.
```

```
    DISPLAY "Enter second number (1 digit) : " WITH NO ADVANCING.
```

```
    ACCEPT Num2.
```

```
    MULTIPLY Num1 BY Num2 GIVING Result.
```

```
    DISPLAY "Result is = ", Result.
```

```
    STOP RUN.
```

Ада

```
-- file: math_functions.adb

package body Math_Functions is

    Epsilon : constant := 1.0e-6;

    function Sqrt (X : Float) return Float is
        Result : Float := X / 2.0;
    begin
        while abs (Result * Result - X) > Epsilon loop
            Result := 0.5 * (X / Result + Result);
        end loop;
        return Result;
    end Sqrt;

    function Exp (Base : Float; Exponent : Float) return Float is
    begin
        -- need implementation code here
        return 1.0;
    end Exp;

end Math_Functions;
```

Функциональные языки

- Декларативные
- Все есть функция
- Суперпозиция функций
- Переменных нет
- Кроме функций есть еще и списки
- В дикой природе практически не встречаются, если же даже встречаются, то на них никто ничего серьезного не пишет, а если и пишет, то нужно это только для защиты диссертаций по дискретной математике

Функциональные языки

- LISP — (Джон МакКарти, 1958)
 - Scheme
 - Clojure
 - Common Lisp
- Erlang — (Joe Armstrong, 1986) функциональный язык с поддержкой **процессов**.
- APL — предшественник современных научных вычислительных сред, таких как MATLAB.
- ML (Робин Милнер, 1979, из ныне используемых диалектов известны Standard ML и Objective CAML).
- F# — функциональный язык семейства ML для платформы .NET
- Scala
- Miranda (Дэвид Тёрнер, 1985, который впоследствии дал развитие языку Haskell).
- Nemerle — гибридный функционально/императивный язык.
- XSLT и XQuery
- Haskell — чистый функциональный. Назван в честь Хаскелла Карри.

LISP

```
; LISP
```

```
(defun fibonacci(n)
  (if (or (= n 0) (= n 1))
      1
      (+ (fibonacci (- n 1))
          (fibonacci (- n 2)))))
```

HASKELL

```
trapezeIntegrate f a b n =  
    ((sum $ map f [a + h, a + 2*h .. b - h]) + t) * h  
  where  
    t = (f a + f b)/2  
    h = (b - a) / n  
  
main = do  
    print $ trapezeIntegrate (\x -> x*sin x) 0 (2*pi) 100
```

HASKELL

```
let_in =let in'let'in=let in let in" let" in
      let in let
let'in let_ _in =
  let_>>_in in
  in'let'in++
    let in_let'in=let
      in " let in let" in let'in
in_let'in in'let'in
```

```
fix$(<$><$>(:)<*>((<$>((:[{- Jörð -}])<$>))(<=<><$>(*)<$>(>>=)(+)($))$1
```

Объектно-ориентированные языки

- В чистом виде не встречаются
- В основе лежит взаимодействие объектов через сообщения
- В основе лежит класс или прототип
- Объект – есть экземпляр класса
- Инкапсуляция, наследование и полиморфизм

Объектно-ориентированные языки

- Почти все современные языки программирования являются объектно-ориентированными в том или ином виде
- Да и процедурными тоже
- И функциональные возможности есть тоже у многих, поэтому...

SMALLTALK

```
exampleWithNumber: x
  | y |
  true & false not & (nil isNil) ifFalse: [self halt].
  y := self size + super size.
  #($a #a "a" 1 1.0)
    do: [ :each |
      Transcript show: (each class name);
      show: ' ' ].
  ^x < y
```

... подавляющее большинство современных
ЯЗЫКОВ

МУЛЬТИПАРАДИГМЕННЫЕ

- Структурное программирование
- Функциональное программирование
- Логическое программирование
- Агент-ориентированное программирование
- Субъективно-ориентированное программирование
- Обобщённое программирование
- Доказательное программирование
- Порождающее программирование
- Аспектно-ориентированное программирование
- Автоматное программирование
- Событийно-ориентированное программирование
- Компонентно-ориентированное программирование
- Грамотное программирование (Literate Programming)

Очень короткий перечень...

- C++
- Java
- Delphi
- Go
- D
- Objective-C
- Swift
- .NET
- Erlang
- Perl
- Python
- Scala
- ActionScript
- JavaScript
- Ruby
- Dart
- Vala
- PHP

По виду трансляции

- Трансляция – процесс перевода исходного кода на определенном ЯП в машинный код
- Виды трансляции
 - Компиляция
 - Интерпретация
 - Комбинированная (быстрая компиляция, JIT-компиляция, виртуальная машина)

Компиляция

- Компиляция (объектный код) → Сборка (машинный код)
→ Загрузка ОС
- Этапы компиляции
 - Лексический анализ
 - Грамматический разбор (парсинг, результат – дерево разбора)
 - Семантический анализ
 - Оптимизация
 - Кодогенерация

Интерпретация

- Последовательный просмотр исходного кода строка за строкой
- Прямой перевод инструкций на языке программирования в машинные инструкции
- Возможность работы в режиме read-eval-print loop (REPL) – удобно для тестирования идей и обучения

Достоинства

- Простая переносимость
- Меньше кода

Недостатки

- Для работы нужен интерпретатор
- Скорость работы всегда меньше
- Практически нет оптимизации

**Достоинства и недостатки
компиляции абсолютно
симметричны**

**Поэтому большинство
современных языков
используют оба подхода к
трансляции**

Типизация

- Статическая

`int i = 0;`

- Динамическая

`i = 0`

- Строгая

`a = 0`

`b = 'строка'`

`c = a + b` # ошибка

- Слабая

Как и все в мире компьютеров, типизация тоже бывает комбинированная, например статически типизированные языки Delphi и Haskell могут использовать динамическую типизацию через специальные средства, а ЯП Go делает это «из коробки», т. е. при декларированной статической типизации, на этапе компиляции происходит динамическое разрешение типа, то есть можно ввести еще один критерий типизации:

ЯВНАЯ и НЕЯВНАЯ

JavaScript	-	Динамическая		Слабая		Неявная
Ruby	-	Динамическая		Сильная		Неявная
Python	-	Динамическая		Сильная		Неявная
Java	-	Статическая		Сильная		Явная
PHP	-	Динамическая		Слабая		Неявная
C	-	Статическая		Слабая		Явная
C++	-	Статическая		Слабая		Явная
Perl	-	Динамическая		Слабая		Неявная
Objective-C	-	Статическая		Слабая		Явная
C#	-	Статическая		Сильная		Явная
Haskell	-	Статическая		Сильная		Неявная
Common Lisp	-	Динамическая		Сильная		Неявная
D	-	Статическая		Сильная		Явная
Go	-	Статическая		Сильная		Явная
Delphi	-	Статическая		Сильная		Явная

- Варианта статическая слабая неявная не существует
- Зато существует огромное количество языков вообще без типизации

**Отдельно от классификации
рассматриваются
ассемблеры или
*машинный код***

Классификация по назначению

- Общего назначения и специальные
- Примеры специальных языков:
 - SQL (декларативный безтиповый)
 - ISO-7bit
 - Все языки разметки и сериализации (XML, HTML, YAML, JSON и т. д.)
 - CSS
 - Регулярные выражения
 - Языки описания оборудования (VHDL, SystemVerilog)

Эзотерические ЯП

- INTERCAL-подобные. Основная идея — максимальное отличие от существующих языков
 - FALSE
- Brainfuck-подобные. Ориентированы на сокращение синтаксиса (оригинальный Brainfuck имеет 8 команд) при сохранении тьюринг-полноты
 - CaneCode
 - Ook! (язык орангутанов)
 - COW (язык парнокопытных)
 - Brainfork (многозадачный Brainfuck)
 - f*ckf*ck
 - DoubleFuck
 - Whitespace
 - Spoon
 - LOLCODE

Эзотерические ЯП

- Использующие многомерные представления программ
 - Byter (двумерный)
 - Befunge (двумерный)
 - Befunge-93 (двумерный, не тьюринг-полный)
 - Unefunge (одномерный)
 - Trefunge (трёхмерный)
 - 4DL (четырёхмерный)
 - Piet (с цветовым кодированием)

Эзотерические ЯП

- И еще много разных, включая языки с нечеловеческой логикой (Var'aq – логика расы клингонов из сериала «Звёздный путь»), языки со стихотворным синтаксисом и языки, которые созданы специально, чтобы запутать программиста
- Раньше я думал, что это Perl, но... вот программ на ЯП **Malbolge**

```
( ' & % : 9 ] ! ~ } | z 2 V x w v - , P 0 q p o n l $ H j i g  
% e B @ @ > } = < M : 9 w v 6 W s U 2 T | n m - , j c L ( I & % $ # " ` C B ] V ?  
T x < u V t T ` R p o 3 N l F . J h + + F d b C B A @ ? ] ! ~ |  
4 X z y T T 4 3 Q s q q ( L n m k j " F h g $ { z @ >
```

«Hello World» на языке Piet



Спасибо за внимание

- Вопросы?
- Предложения?
- Просьбы?
- Жалобы?