

Технологии программирования

Лекция 2

Базовые понятия о языках программирования

Старичков Н.Ю.

Парадигмы программирования

ИМПЕРАТИВНОЕ /
ДЕКЛАРАТИВНОЕ

Императивное - «приказы»

Императивное программирование

Imperio!

- в исходном коде программы записываются инструкции (команды);
- инструкции должны выполняться последовательно;
- данные, получаемые при выполнении предыдущих инструкций, могут читаться из памяти последующими инструкциями;
- данные, полученные при выполнении инструкции, могут записываться в память

Императивное программирование

Примеры языков

- C/C++
- Java
- Python
- JavaScript

```
// Imperative Programming
let array = [1, 2, 3, 4, 5, 6]
var evenNumbers: [Int] = []

for i in 0..
```

Императивное программирование

- Процедурное программирование
- Структурное программирование
- Аспектно-ориентированное программирование
- Объектно-ориентированное программирование
- Обобщенное программирование

Декларативное - «Описание»

Пример про друзей, идущих в
ГОСТИ

«Выйти на остановке такой-то,
пойти туда, повернуть
направо, красивый дом, зайти
в первый подъезд»»

«Выйти на остановке такой-то,
пойти туда, повернуть
направо, красивый дом, зайти
в первый подъезд»

императивное

«Мой адрес вот такой, как
добраться решишь сам»»

«Мой адрес вот такой, как
добратся решишь сам»

декларативное

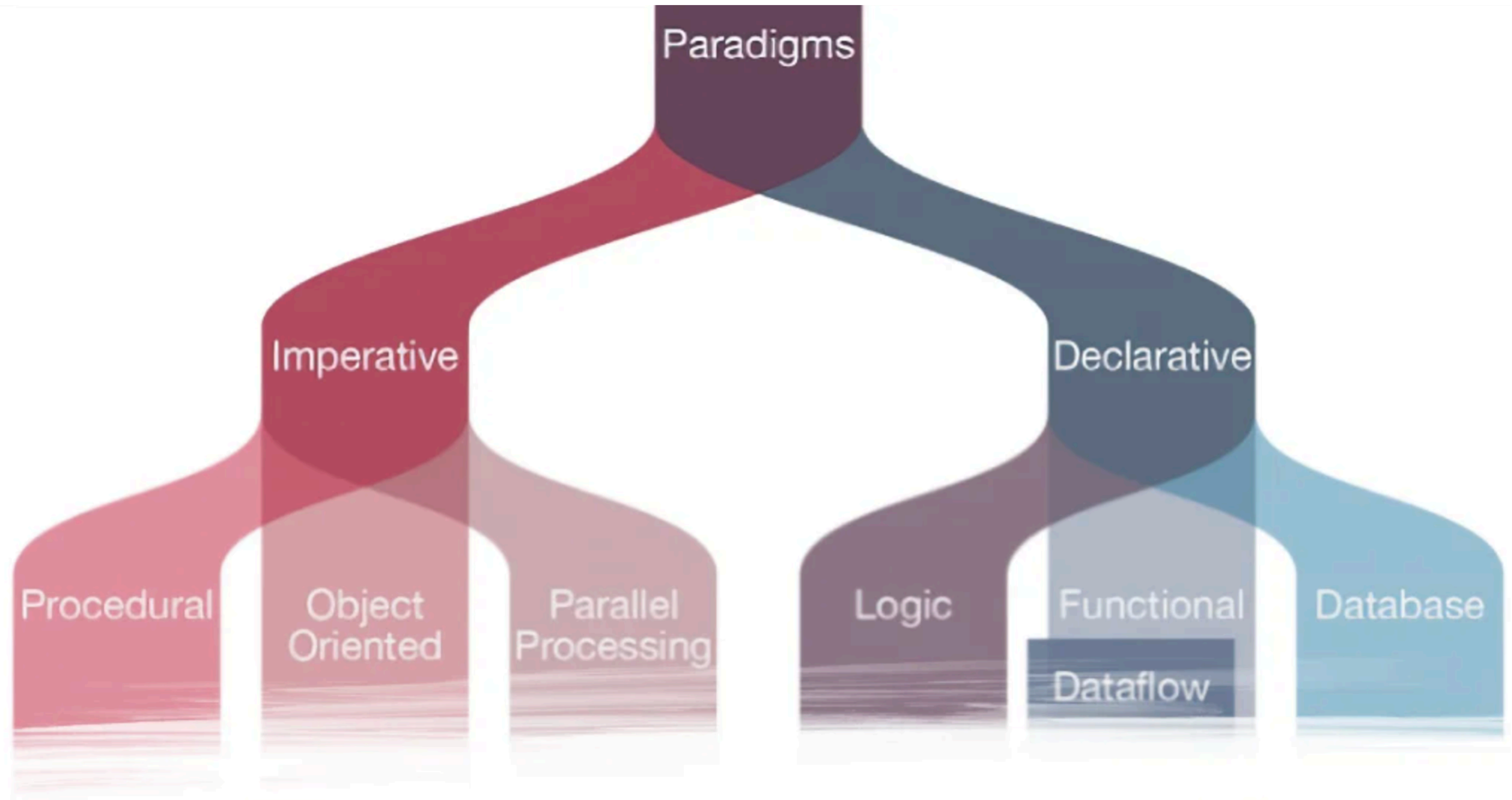
Декларативное программирование

- Логическое программирование
- Функциональное программирование

```
DEFINE
(( (RVRSE, (LAMBDA, (L), (COND, ((NULL, L), NIL),
    (T, (CONS, (RVRSE, (CDR, L)), (CONS, (CAR, L), NIL)))))),
(RVDE, (LAMBDA, (L), (REV, L, NIL))),
(REV, (LAMBDA, (J, K), (COND, ((NULL, J), K),
    (T, (REV, (CDR, J), (CONS, (CAR, J), K)))))))
()
RVRSE ((A,B,C,D,E)) ()
RVDE ((A,B,C,D,E)) ()
```

LISP I Programmer's Manual, 1960

```
loop(Users, N) ->
receive
{connect, Pid, User, Password} ->
io:format("connection request from:~p ~p ~p~n",
    [Pid, User, Password]),
case member({User, Password}, Users) of
true ->
    Max = max_connections(),
    if
        N > Max ->
            Pid ! {ftp_server,
                {error, too_many_connections}},
            loop(Users, N);
        true ->
            New = spawn_link(?MODULE, handler, [Pid]),
            Pid ! {ftp_server, {ok, New}},
            loop(Users, N + 1)
    end;
false ->
    Pid ! {ftp_server, {error, rejected}},
    loop(Users, N)
end;
{'EXIT', Pid} ->
io:format("Handler ~p died~n", [Pid]),
loop(Users, lists:max(N-1, 0));
Any ->
io:format("received:~p~n", [Any]),
loop(Users, N)
end.
```

Source: <https://www.watelectronics.com/types-of-programming-languages-with-differences/>

Метапрограммирование

Метапрограммирование
«программы генерируют
программы»

Язык программирования может
реализовывать несколько
парадигм

C++

ООП и Метапрограммирование

```
13 class Blinker {
14     private:
15         byte pinLED;
16
17         boolean ledState = LOW;
18
19         unsigned long timeLedOn;
20         unsigned long timeLedOff;
21
22         unsigned long nextChangeTime = 0;
23
24     public:
25         Blinker(byte pinLED, unsigned long timeLedOn, unsigned
26             this->pinLED = pinLED;
27             this->timeLedOn = timeLedOn;
28             this->timeLedOff = timeLedOff;
29
30             pinMode(pinLED, OUTPUT);
```

```
template <typename T, typename Arg>
T createT(Arg&& arg){
    return T(std::forward<Arg>(arg));
}

int main(){

    int lvalue{2020};

    //std::unique_ptr<int> uniqZero = std::make_unique<int>(); // (1)
    auto uniqEleven = createT<int>(2011); // (2)
    auto uniqTwenty = createT<int>(lvalue); // (3)
    //auto uniqType = std::make_unique<MyType>(lvalue, 3.14, true); // (4)

}
```

**КОМПИЛИРУЕМЫЕ /
ИНТЕРПРЕТИРУЕМЫЕ**

Компилируемый -
специальная программа
(«компилятор») переводит
текст программы в машинный
код, который уже исполняется

Интерпретируемый -
специальная программа
(«интерпретатор») выполняет
команды из текста программы
по очереди

Примеры языков программирования

- **Компилируемые:**
 - C, C++, Pascal, Rust, GO (компилируются в машинный код)
 - Java, Scala, Kotlin, C# (компилируются в байт-код)
- **Интерпретируемые:**
 - Python, JavaScript, BASIC

ТИПИЗАЦИЯ

Типизация

Сильная (строгая) / слабая

- Сильная
 - $A = 5; B = \text{«name»}; A = B$ **error!**
- Слабая
 - $A = 5; B = \text{«name»}; B = A + B$ ($=\text{«5name»}$)

Типизация

Статическая / динамическая

- **Статическая**
 - Тип переменной известен до момента выполнения программы
- **Динамическая**
 - Тип переменной определяется по ходу выполнения программы

Типизация

Явная / неявная

- Явная
 - `int a = 5;` (C++)
- Неявная
 - `a = 5` (Python)

>>: tbc...