

COMP26020 Programming Languages and Paradigms Part 1: C Programming

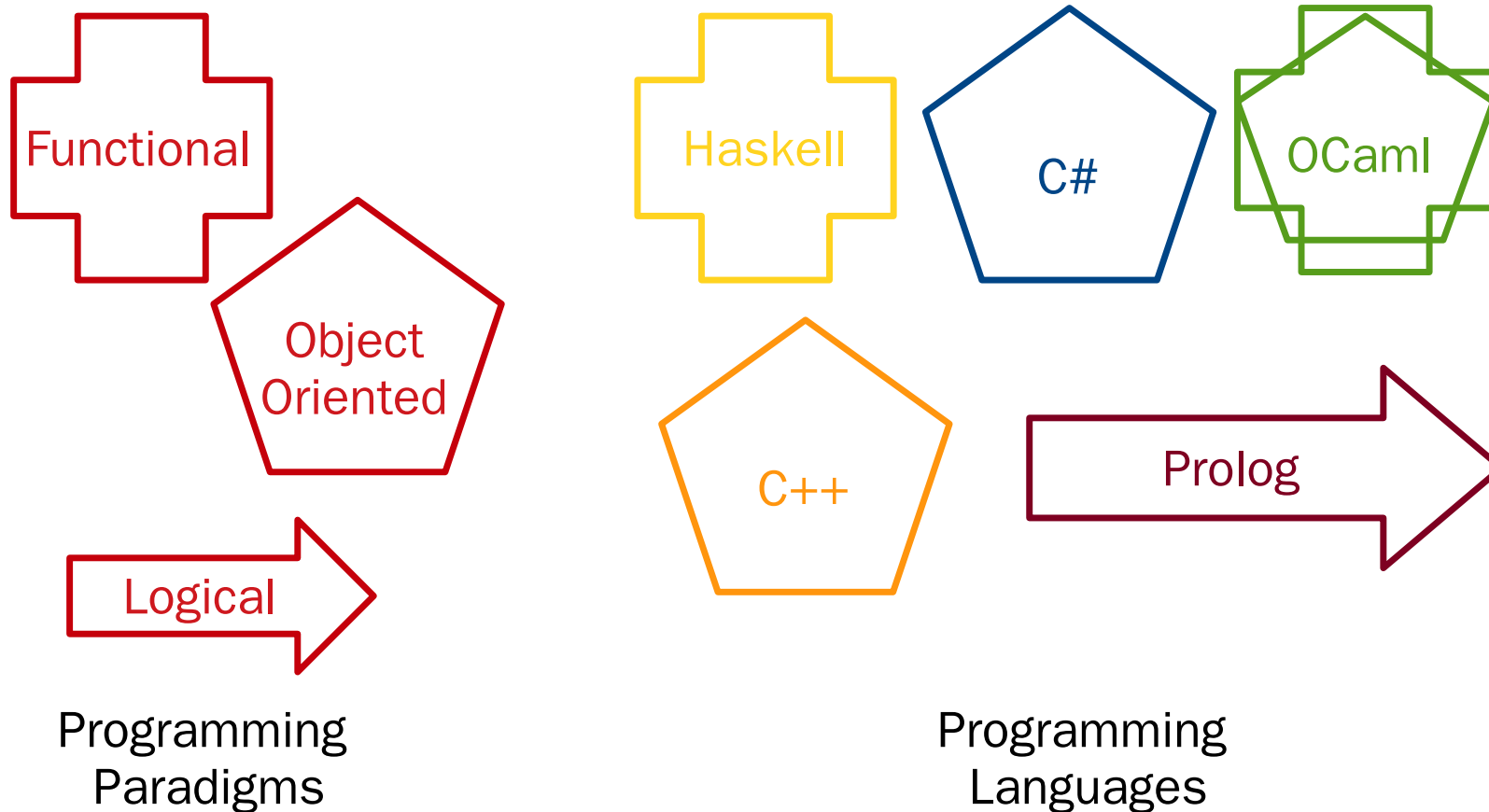
Programming Paradigms

What's a Programming Paradigm?

- A programming paradigm defines a **fundamental style of programming ...**
 - How the programmer can describe the program in the source code:
 - the **data** used by the program, and the **computations** manipulating the data

What's a Programming Paradigm?

- A programming paradigm defines a **fundamental style of programming ...**
 - How the programmer can describe the program in the source code:
 - the **data** used by the program, and the **computations** manipulating the data
- How do programming paradigms related to programming languages?
 - **Paradigms can be used to classify languages** based on the programming styles they support



- **A language belongs to one or several paradigms**, i.e. it makes it easy to develop using the style defined by the paradigm

adapted from <https://www.janeve.me/software-programming/understanding-programming-paradigms>

A Paradigm is a Programming Style

Example: multiplying by 2 each element of an array in JavaScript

A Paradigm is a Programming Style

Example: multiplying by 2 each element of an array in JavaScript

1. Using the **imperative** paradigm:

```
function mult_by_two_imperative (array) {  
  let results = []  
  for (let i = 0; i < array.length; i++){  
    results.push(array[i] * 2)  
  }  
  return results  
}
```

[01-programming-paradigms/mult2-imperative.js](#) 

A Paradigm is a Programming Style

Example: multiplying by 2 each element of an array in JavaScript

1. Using the **imperative** paradigm:

```
function mult_by_two_imperative (array) {  
  let results = []  
  for (let i = 0; i < array.length; i++){  
    results.push(array[i] * 2)  
  }  
  return results  
}
```

[01-programming-paradigms/mult2-imperative.js](#) 

2. Using the **declarative** paradigm:

```
function mult_by_two_declarative (array) {  
  return array.map((item) => item * 2)  
}
```

[01-programming-paradigms/mult2-declarative.js](#) 

Result is the same, the way to describe the computations is different

A Programming Paradigm is a Programming Style

- Paradigm defines how the programmer describes the program **computations**
 - Can be sequences of statements, executed sequentially or in parallel
 - Can be divided into functions
 - Can also be *how the result should look like*
 - etc.

A Programming Paradigm is a Programming Style

- Paradigm defines how the programmer describes the program **computations**
 - Can be sequences of statements, executed sequentially or in parallel
 - Can be divided into functions
 - Can also be *how the result should look like*
 - etc.
- Paradigm also defines how the programmer describes the **data** computations are applied on
 - Basic types
 - Custom data structures
 - Local/global variables
 - etc.

Picking a Programming Paradigm

- Some paradigms are (much) better suited than others to solve a given kind of software engineering problem

Picking a Programming Paradigm

- Some paradigms are (much) better suited than others to solve a given kind of software engineering problem
- Choosing a paradigm and a language to solve a given software engineering problem has **huge consequences in terms of the design and behaviour of the solution**, impacting among others:

Picking a Programming Paradigm

- Some paradigms are (much) better suited than others to solve a given kind of software engineering problem
- Choosing a paradigm and a language to solve a given software engineering problem has **huge consequences in terms of the design and behaviour of the solution**, impacting among others:
 - Code structure and modularity, clarity/complexity to understand, amount of LoC

Picking a Programming Paradigm

- **Some paradigms are (much) better suited than others to solve a given kind of software engineering problem**
- Choosing a paradigm and a language to solve a given software engineering problem has **huge consequences in terms of the design and behaviour of the solution**, impacting among others:
 - Code structure and modularity, clarity/complexity to understand, amount of LoC
 - Reusability of code (e.g. OOP's inheritance)

Picking a Programming Paradigm

- Some paradigms are (much) better suited than others to solve a given kind of software engineering problem
- Choosing a paradigm and a language to solve a given software engineering problem has **huge consequences in terms of the design and behaviour of the solution**, impacting among others:
 - Code structure and modularity, clarity/complexity to understand, amount of LoC
 - Reusability of code (e.g. OOP's inheritance)
 - State (memory, variables/objects, data) management

Picking a Programming Paradigm

- Some paradigms are (much) better suited than others to solve a given kind of software engineering problem
- Choosing a paradigm and a language to solve a given software engineering problem has **huge consequences in terms of the design and behaviour of the solution**, impacting among others:
 - Code structure and modularity, clarity/complexity to understand, amount of LoC
 - Reusability of code (e.g. OOP's inheritance)
 - State (memory, variables/objects, data) management
 - If and how parallelism can be expressed

Picking a Programming Paradigm

- **Some paradigms are (much) better suited than others to solve a given kind of software engineering problem**
- Choosing a paradigm and a language to solve a given software engineering problem has **huge consequences in terms of the design and behaviour of the solution**, impacting among others:
 - Code structure and modularity, clarity/complexity to understand, amount of LoC
 - Reusability of code (e.g. OOP's inheritance)
 - State (memory, variables/objects, data) management
 - If and how parallelism can be expressed
 - Which errors are handled and how (error codes, exception, undefined behaviour)

Picking a Programming Paradigm

- **Some paradigms are (much) better suited than others to solve a given kind of software engineering problem**
- Choosing a paradigm and a language to solve a given software engineering problem has **huge consequences in terms of the design and behaviour of the solution**, impacting among others:
 - Code structure and modularity, clarity/complexity to understand, amount of LoC
 - Reusability of code (e.g. OOP's inheritance)
 - State (memory, variables/objects, data) management
 - If and how parallelism can be expressed
 - Which errors are handled and how (error codes, exception, undefined behaviour)
 - Compile-/run-time characteristics such as memory footprint and program speed

Wrapping Up

- Programming paradigm → programming style
 - Each programming language implements one or more paradigms
 - Choose the right paradigm for the right software engineering problem
-

Feedback form: <https://bit.ly/37p8WZ3>

