Joshua Light
.NET Developer

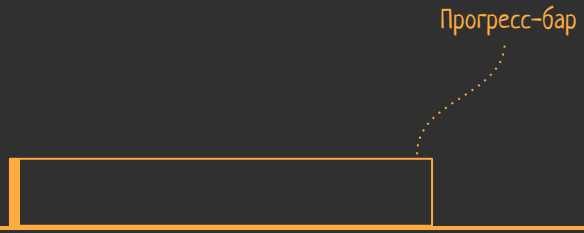
Functional C#

Тут обычно находится слайд, который я использую для тестирования пульта, тем самым загоняя самого себя и свои презентации в рамки примитивной формы, так что с этого момента провозглашается новая эпоха, лишённая недостатков предсказуемого порядка вещей...

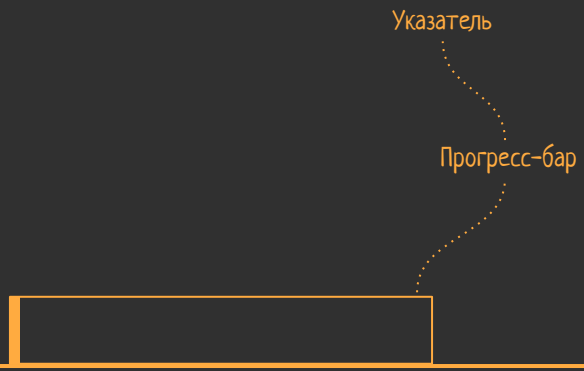
Тут тоже бывает слайд, который я использую для тестирования пульта, тем самым загоняя самого себя и свои презентации в рамки примитивной формы, так что с этого момента провозглашается новая эпоха, лишённая недостатков предсказуемого порядка вещей...

История

История

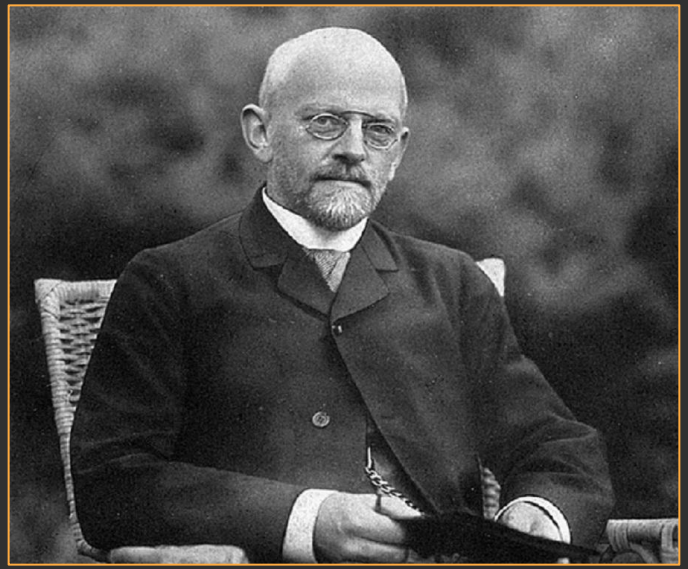


История



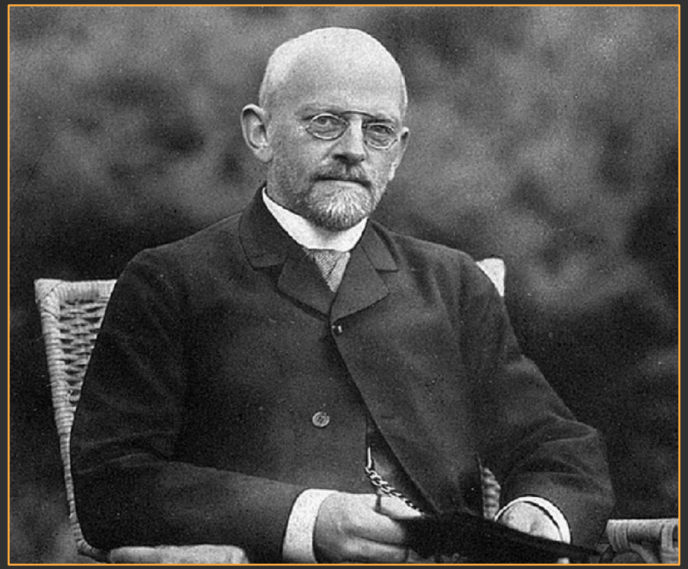
Entscheidungsproblem

Entscheidungsproblem (decision problem)



Entscheidungsproblem (decision problem)

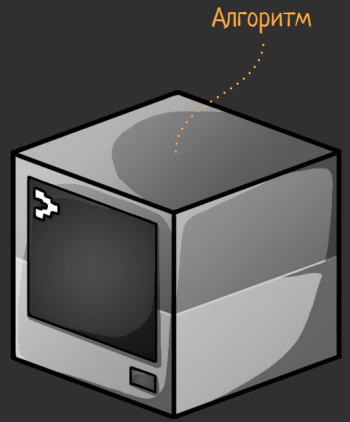
David Hilbert (1862 - 1943)



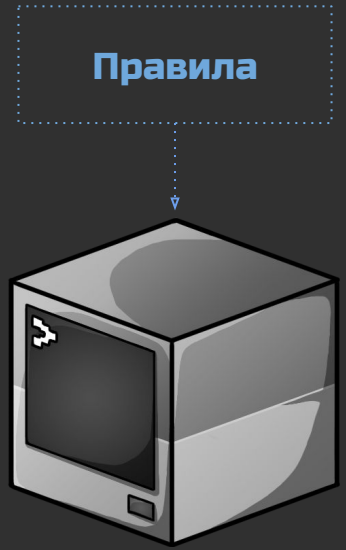
Entscheidungsproblem (decision problem)

Entscheidungsproblem (decision problem)

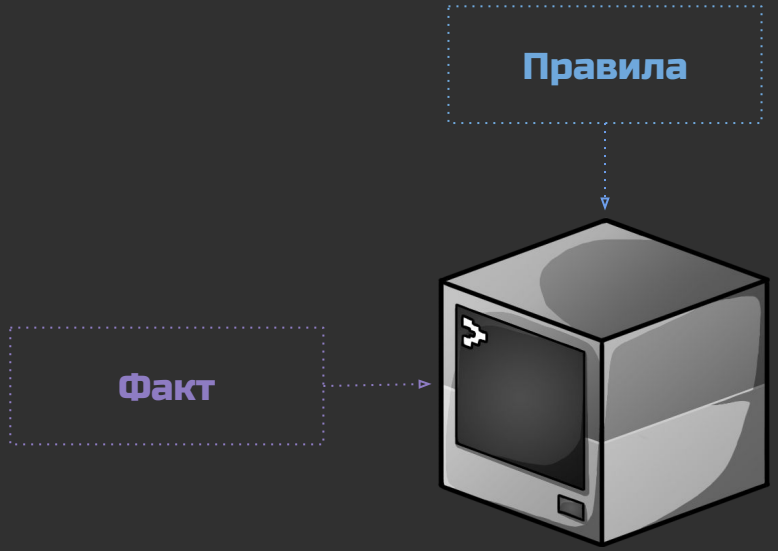
Entscheidungsproblem (decision problem)



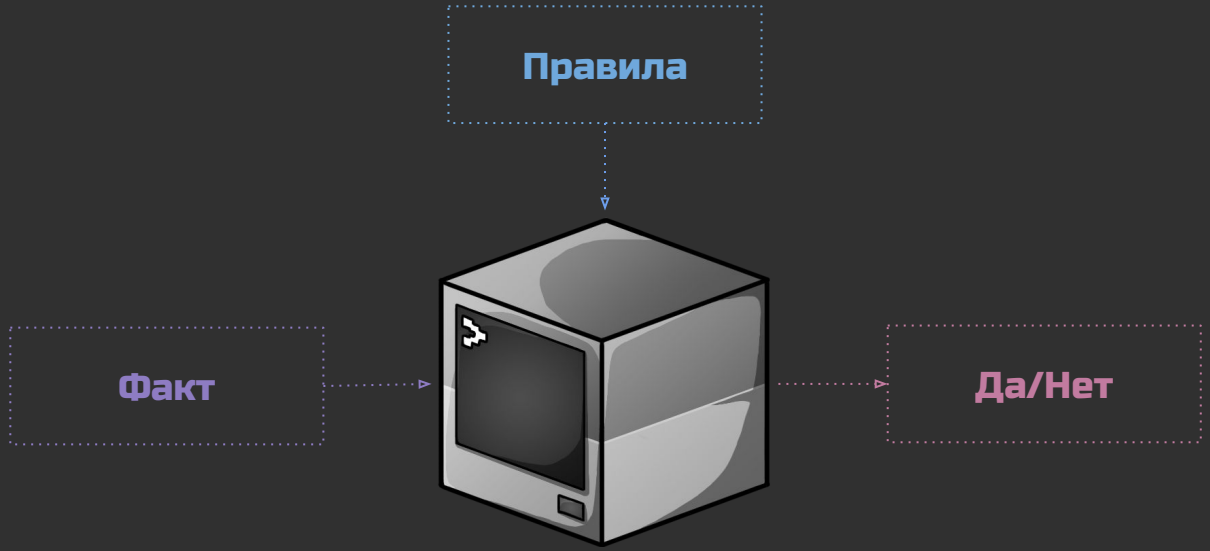
Entscheidungsproblem (decision problem)



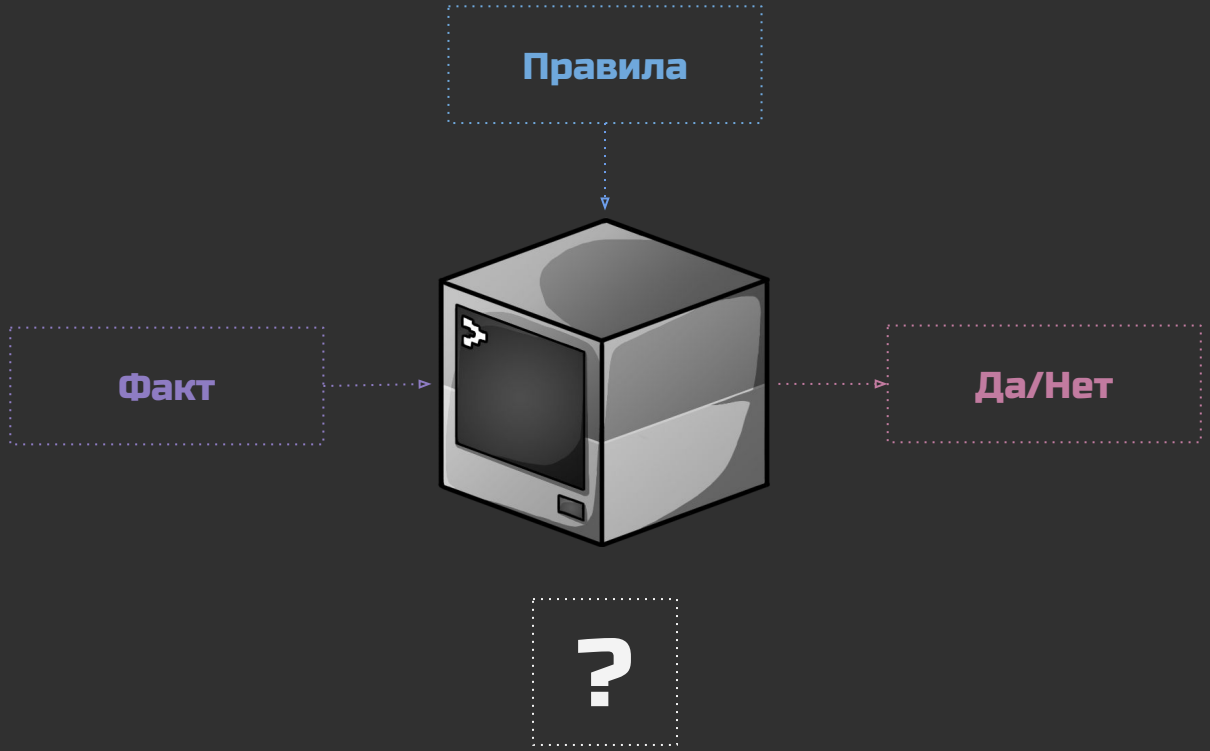
Entscheidungsproblem (decision problem)



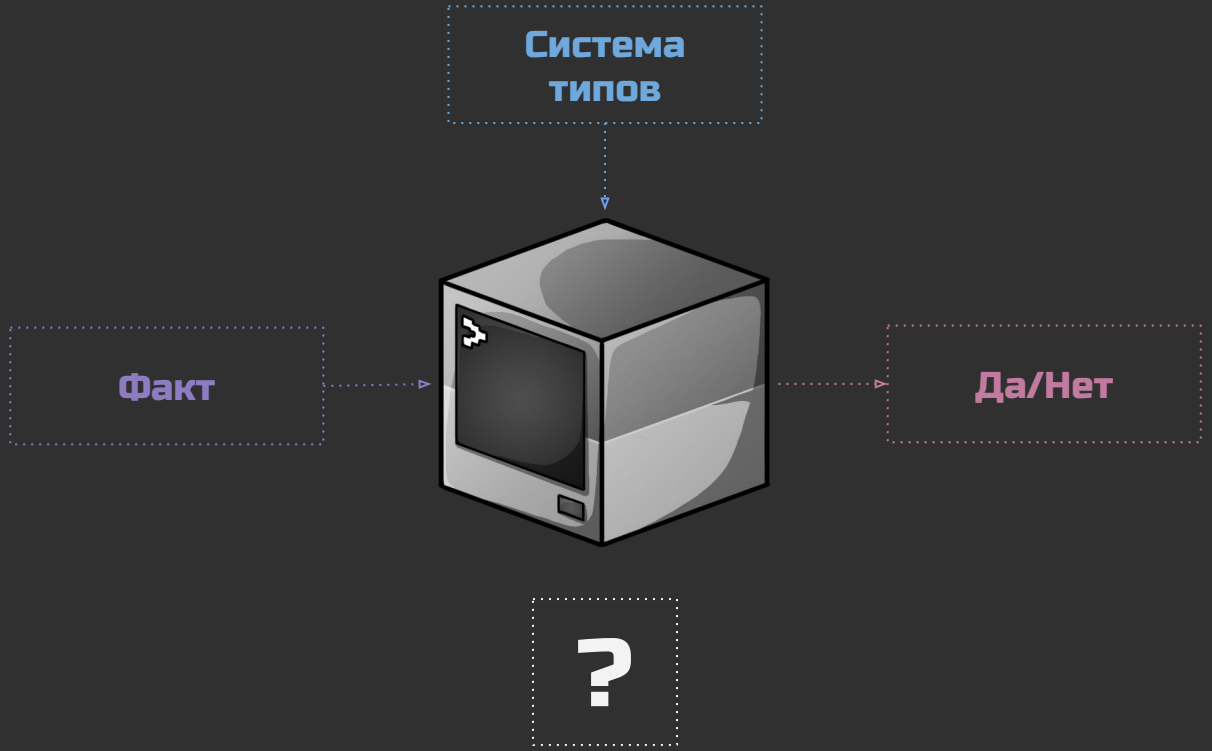
Entscheidungsproblem (decision problem)



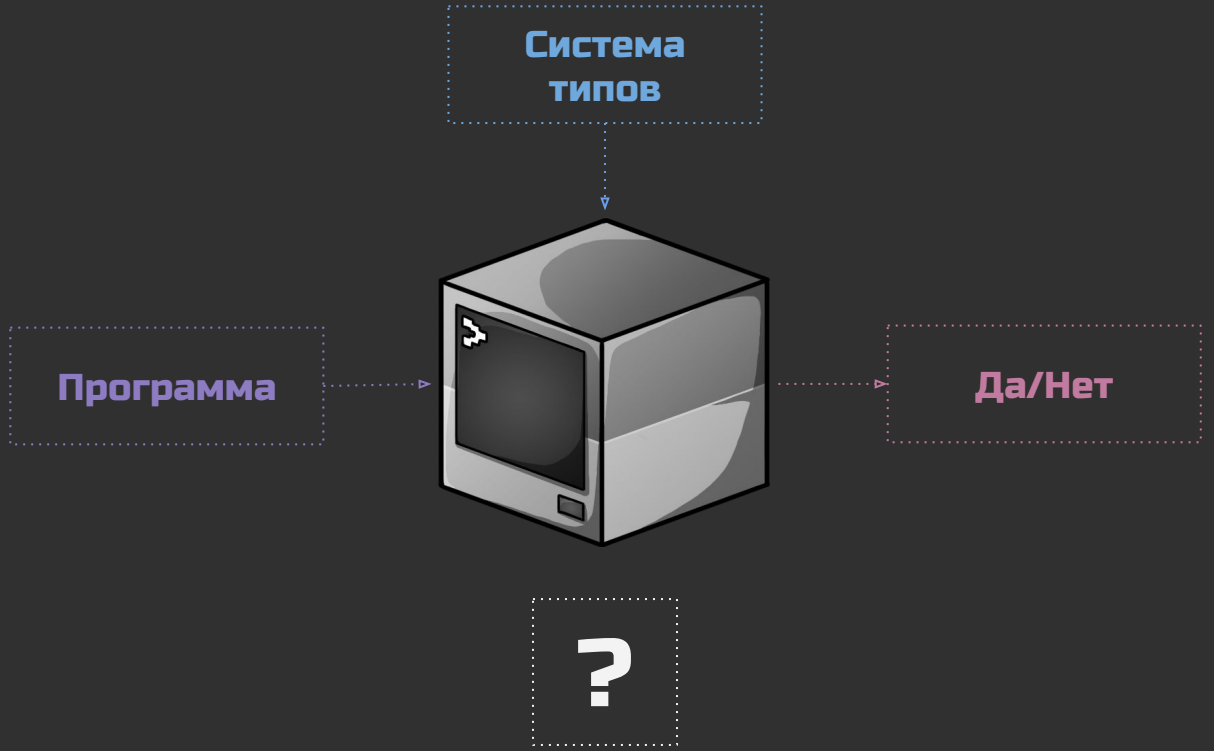
Entscheidungsproblem (decision problem)



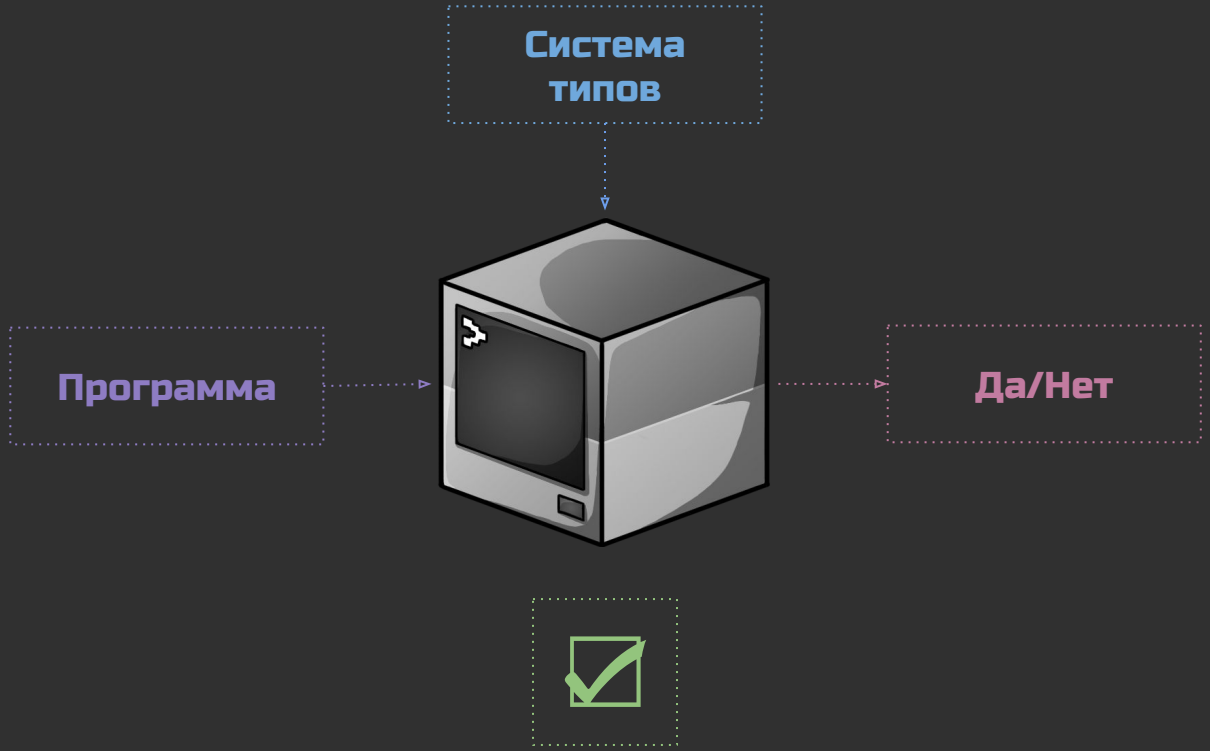
Entscheidungsproblem (decision problem)



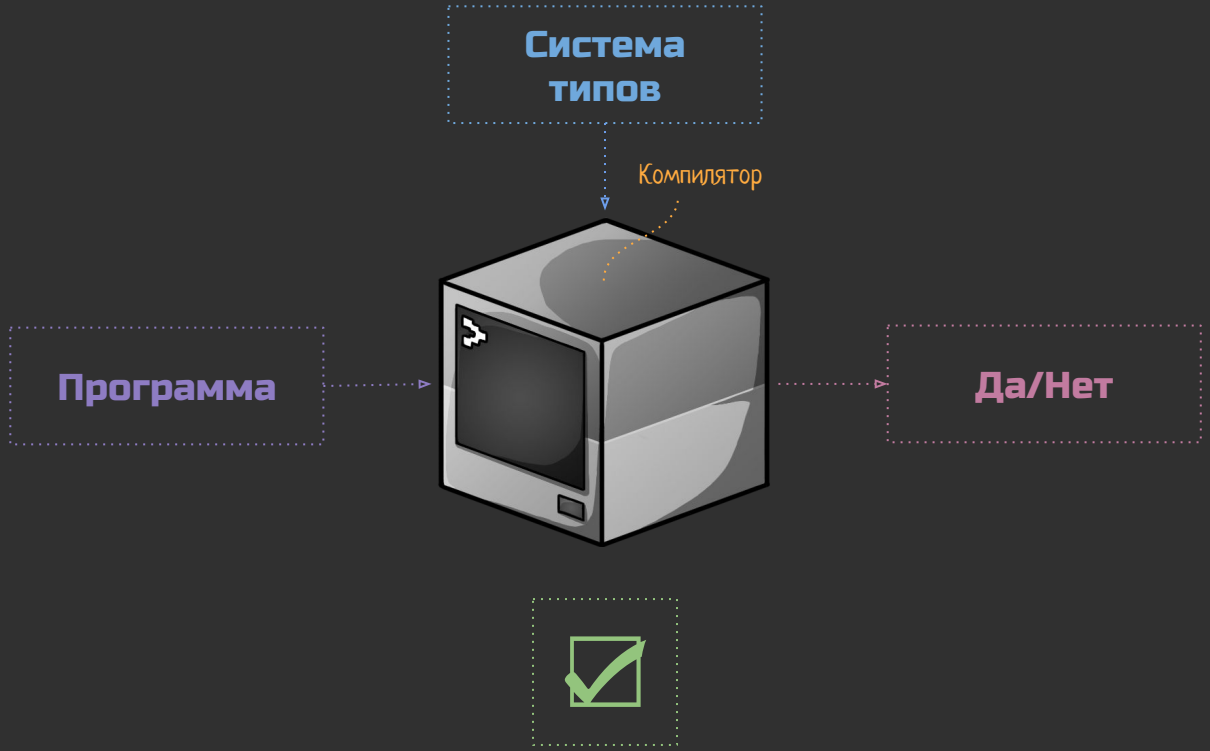
Entscheidungsproblem (decision problem)



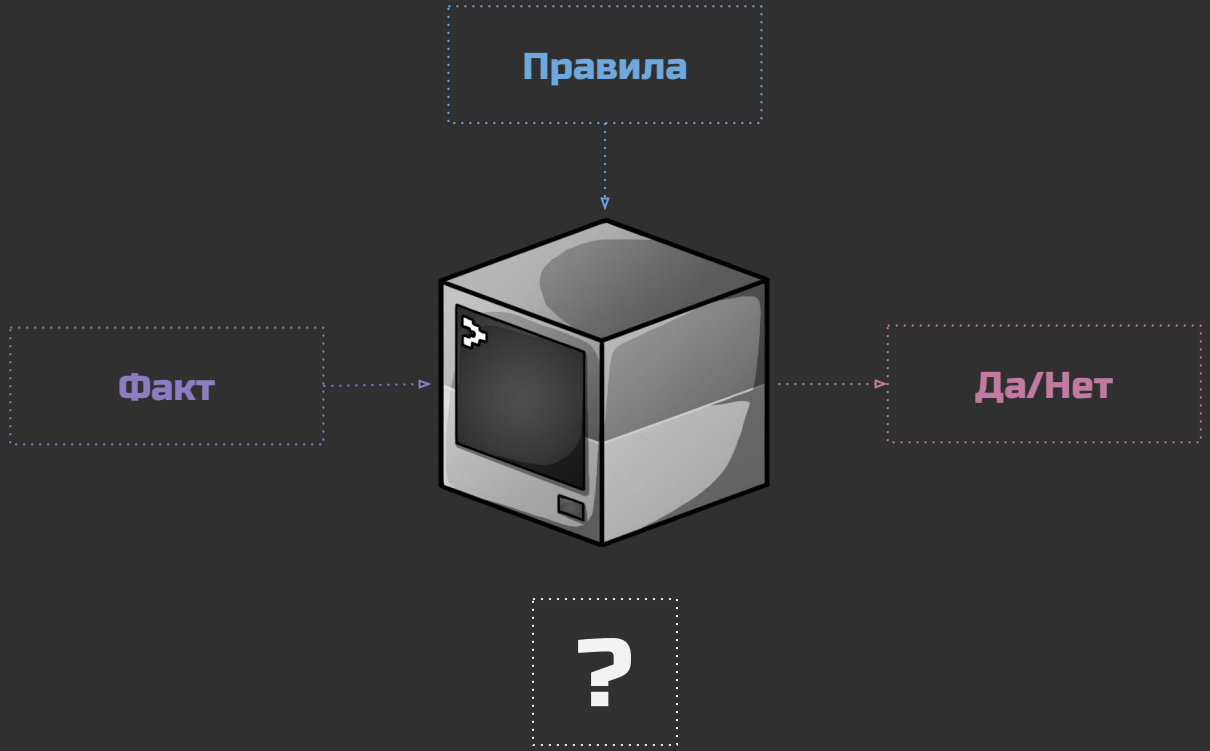
Entscheidungsproblem (decision problem)



Entscheidungsproblem (decision problem)



Entscheidungsproblem (decision problem)



Entscheidungsproblem (decision problem)

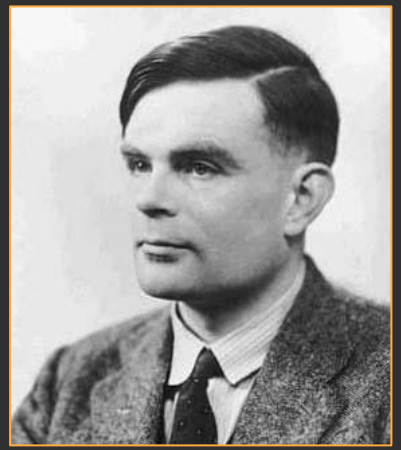
Entscheidungsproblem (decision problem)

Entscheidungsproblem (decision problem)

Alonzo Church (1903 – 1995)

Entscheidungsproblem (decision problem)

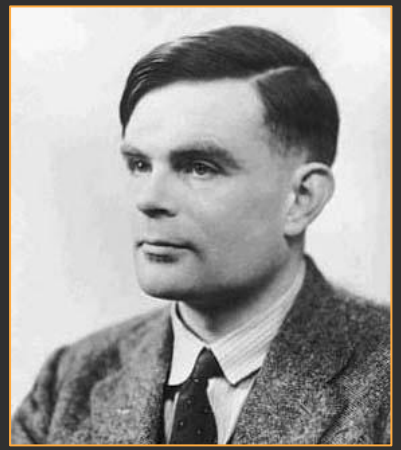
Alonzo Church (1903 – 1995)



Entscheidungsproblem (decision problem)

Alonzo Church (1903 - 1995)

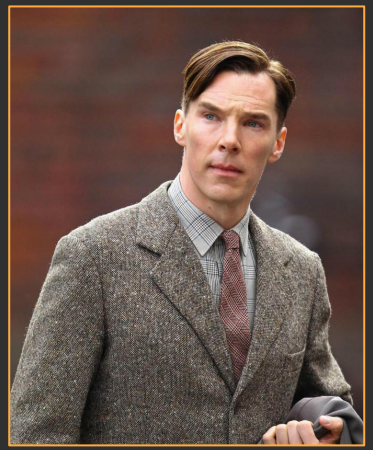
Alan Turing (1912 - 1954)



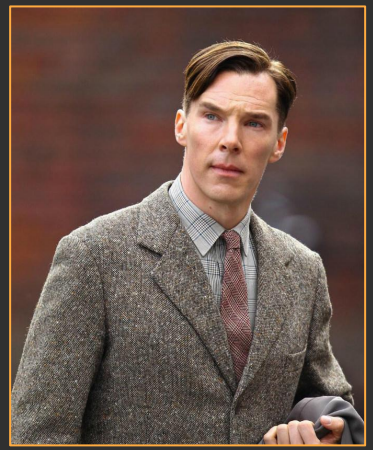
Entscheidungsproblem (decision problem)

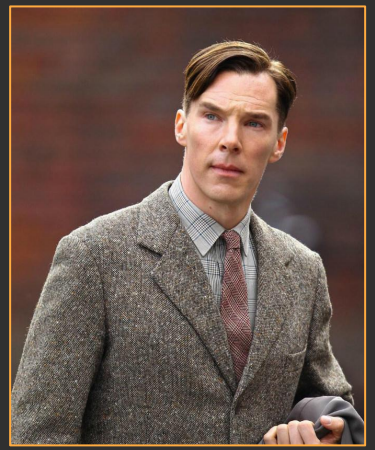
Alonzo Church (1903 - 1995)

Alan Turing (1912 - 1954)

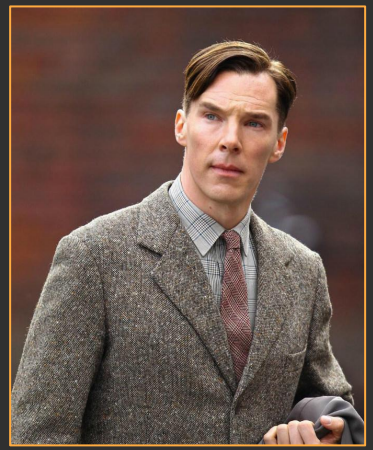


Entscheidungsproblem (decision problem)

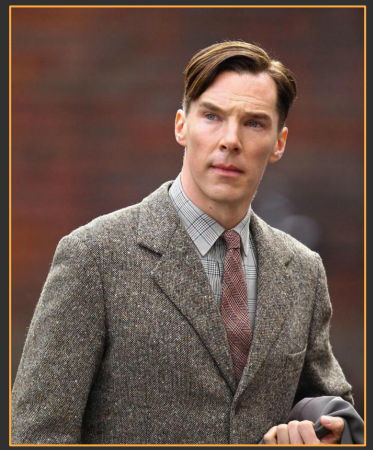


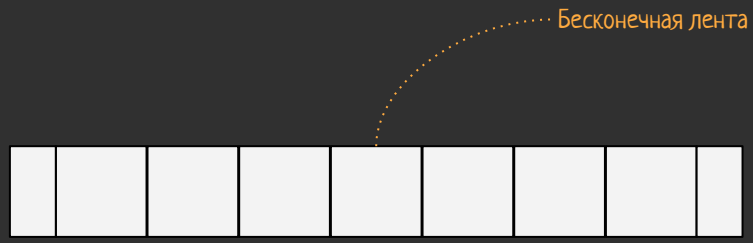


Turing’s Universal Machine

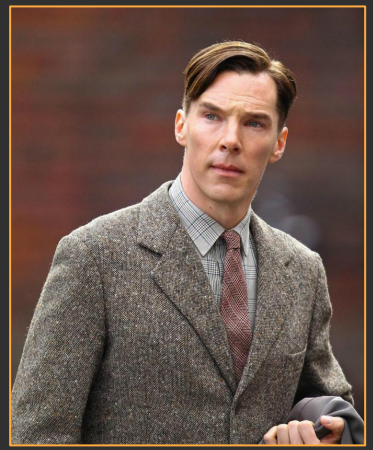


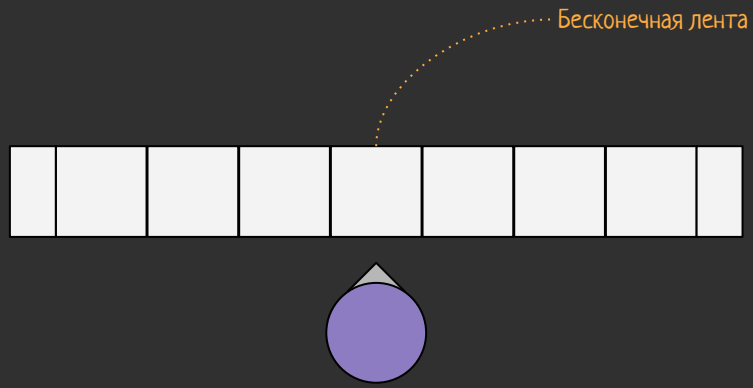
Turing’s Universal Machine



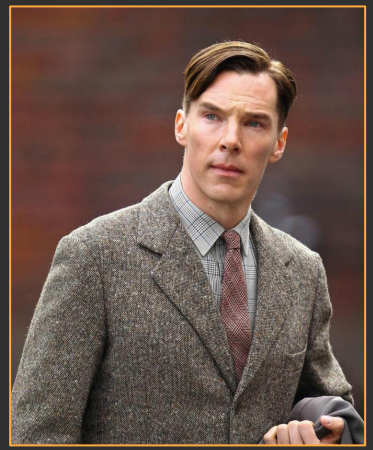


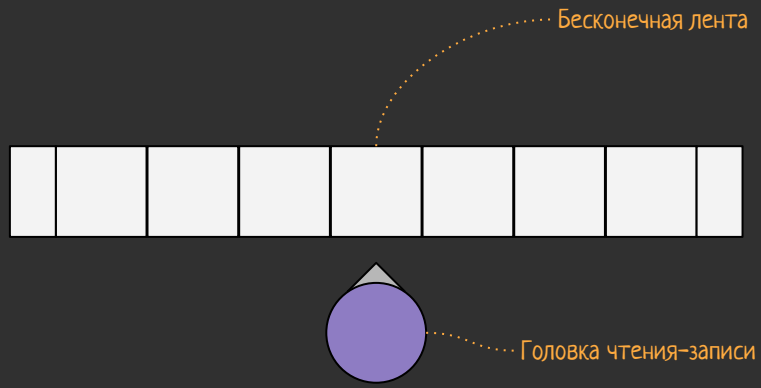
Turing’s Universal Machine



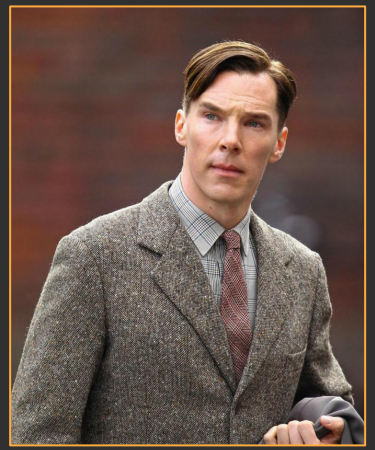


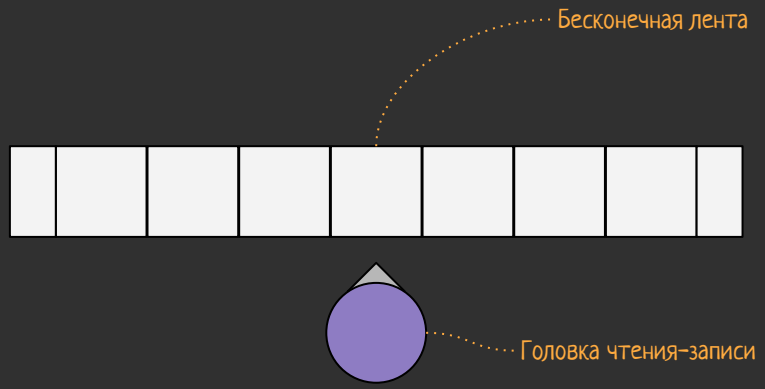
Turing's Universal Machine





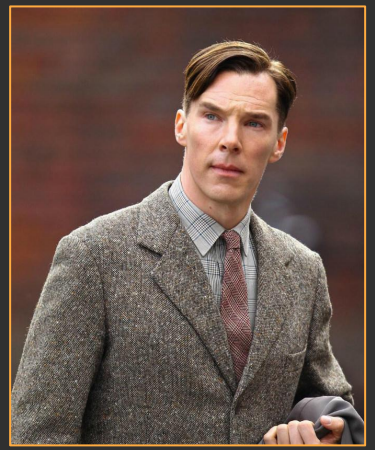
Turing's Universal Machine

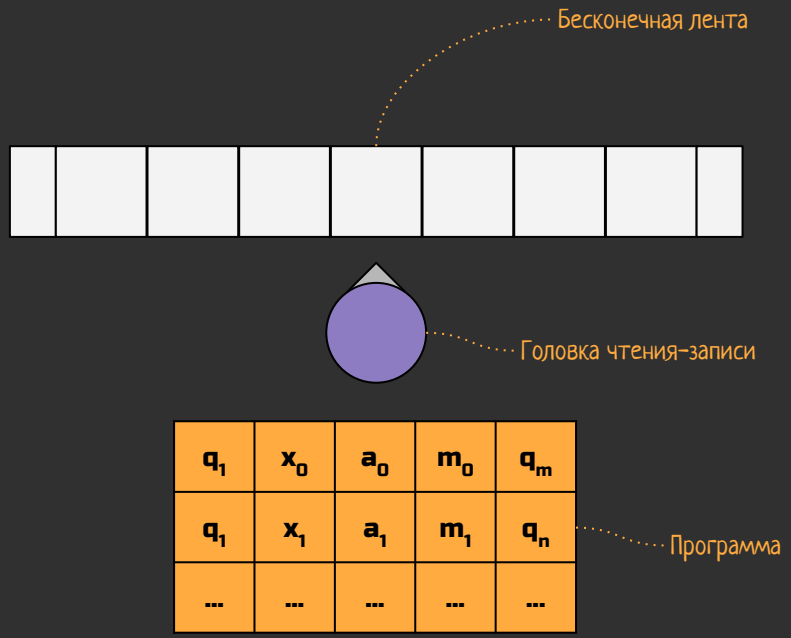




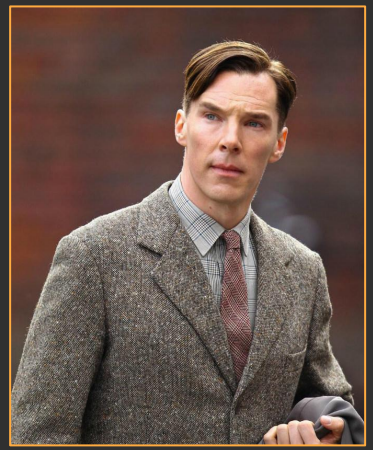
q_1	x_0	a_0	m_0	q_m
q_1	x_1	a_1	m_1	q_n
...	...	...	...	...

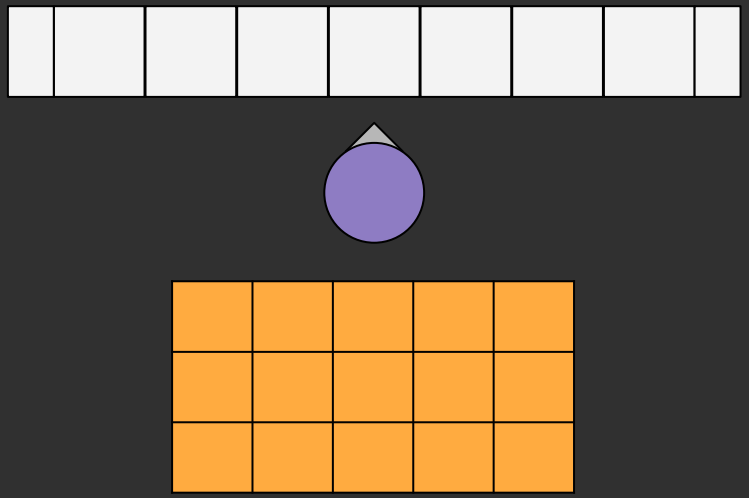
Turing's Universal Machine



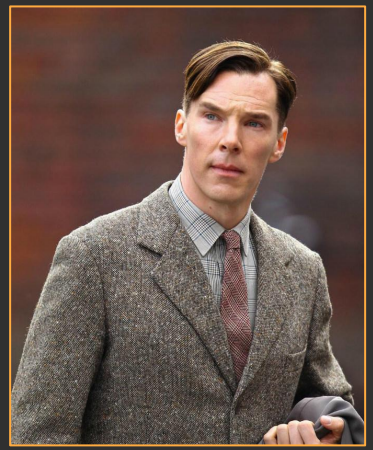


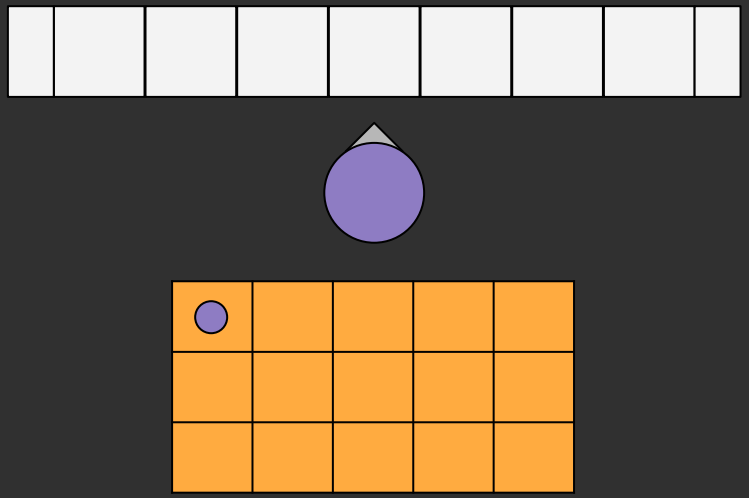
Turing's Universal Machine



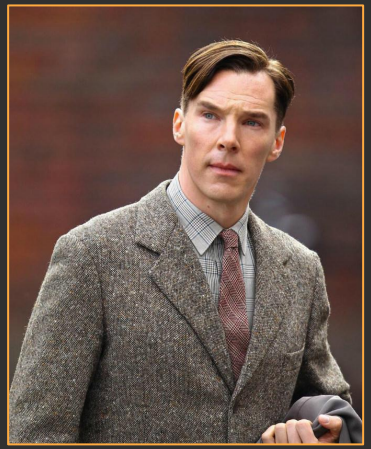


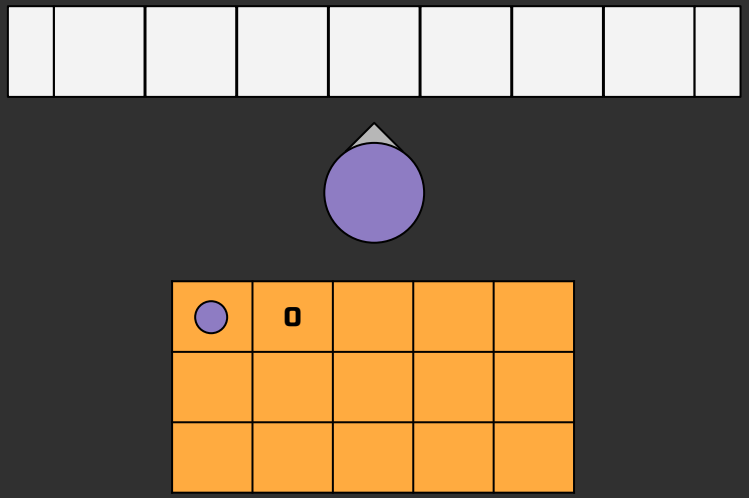
Turing’s Universal Machine



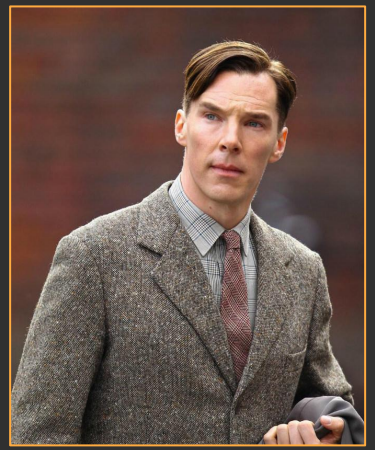


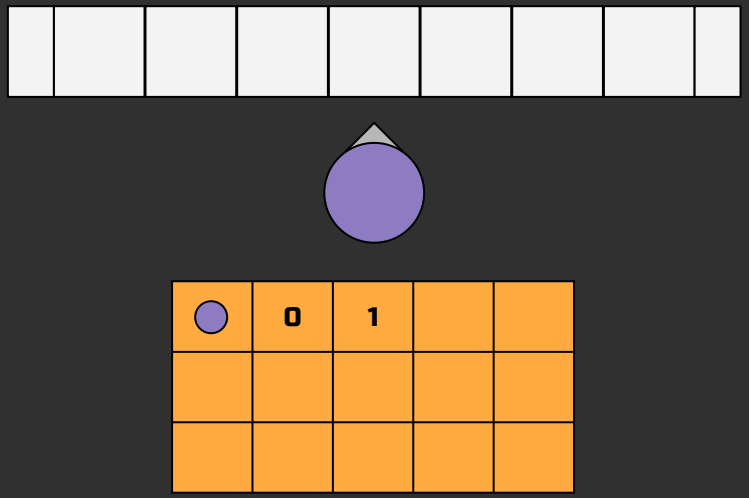
Turing’s Universal Machine



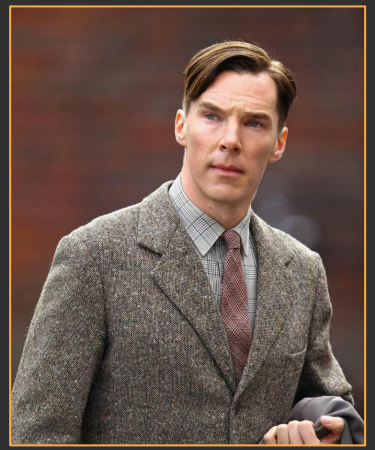


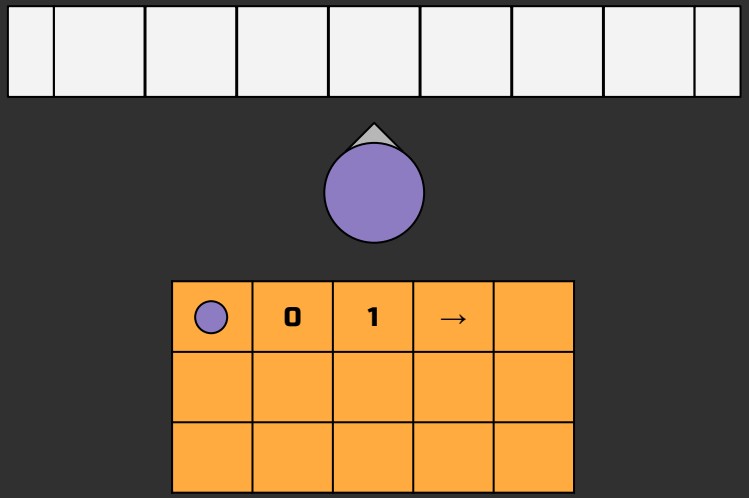
Turing’s Universal Machine



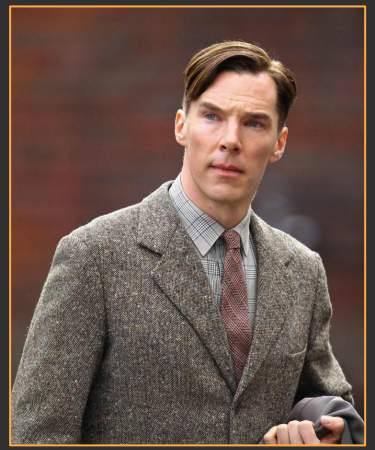


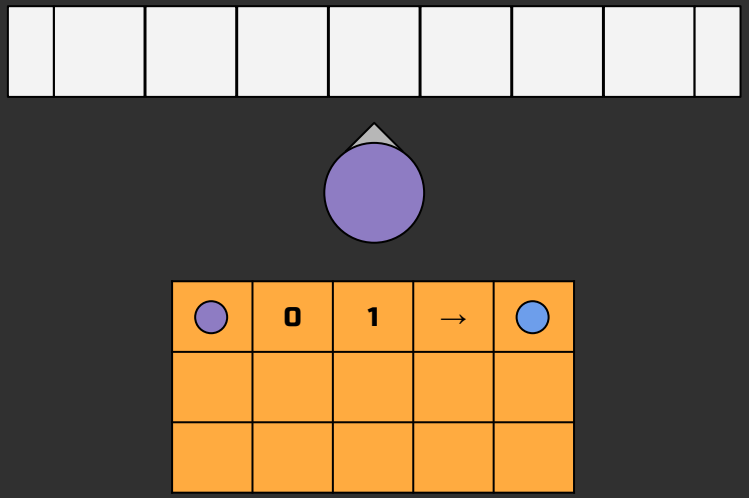
Turing’s Universal Machine



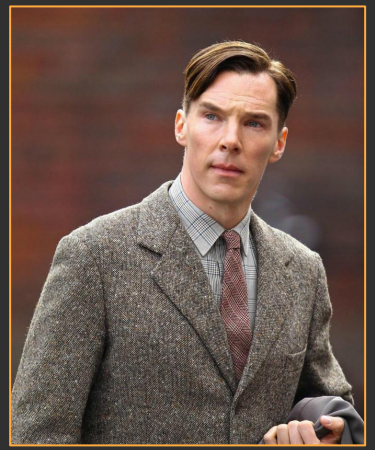


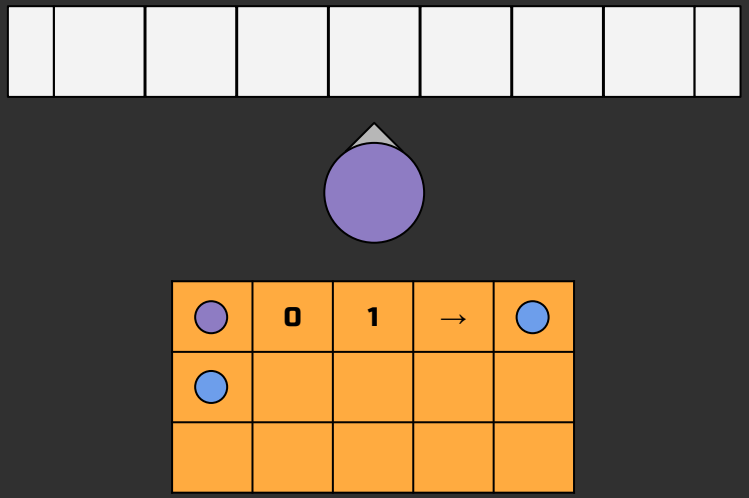
Turing’s Universal Machine



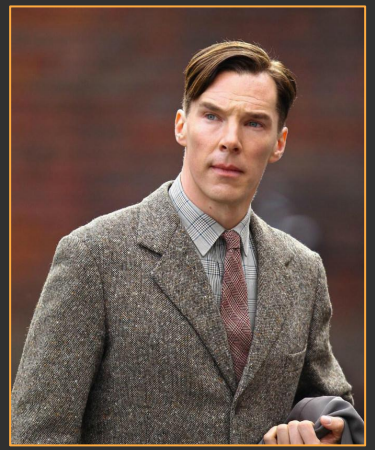


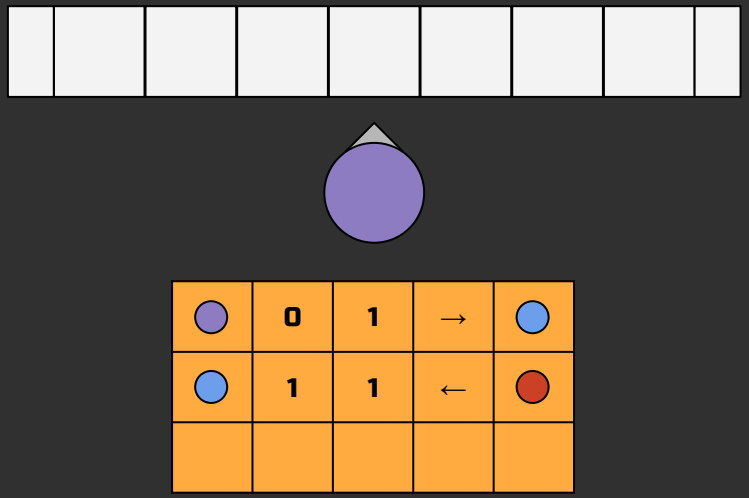
Turing’s Universal Machine



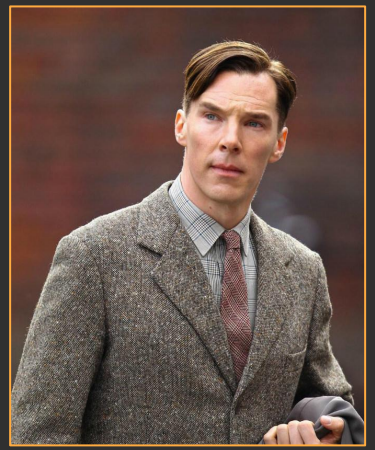


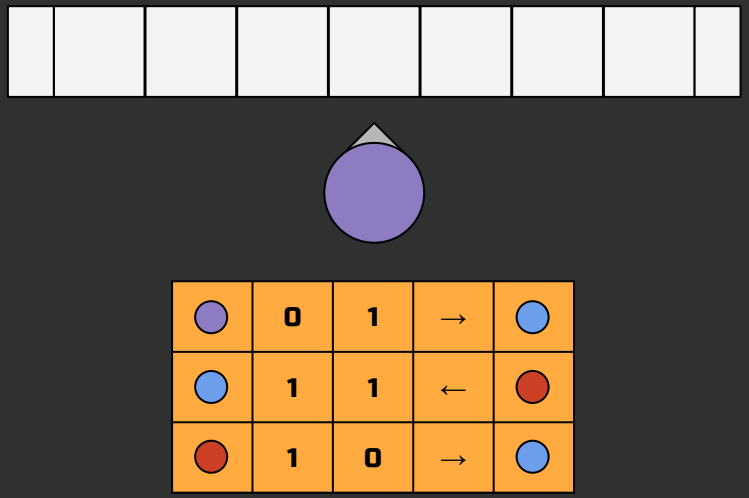
Turing’s Universal Machine



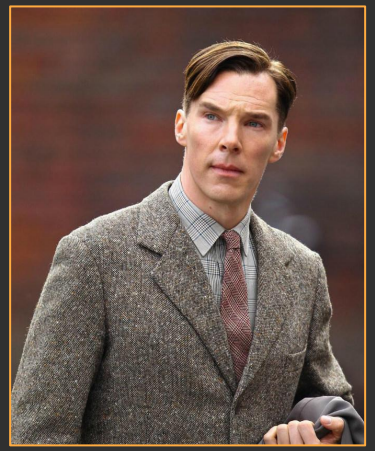


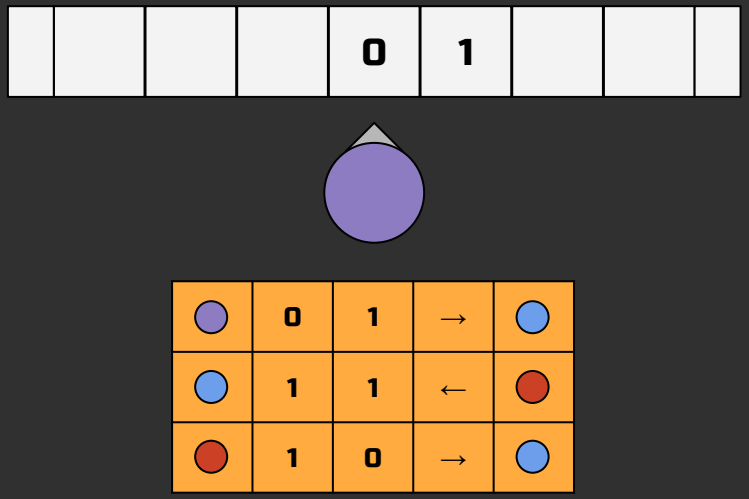
Turing’s Universal Machine



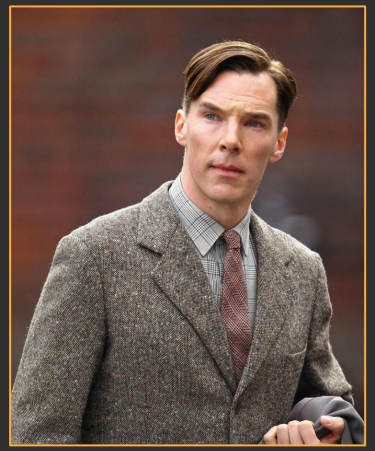


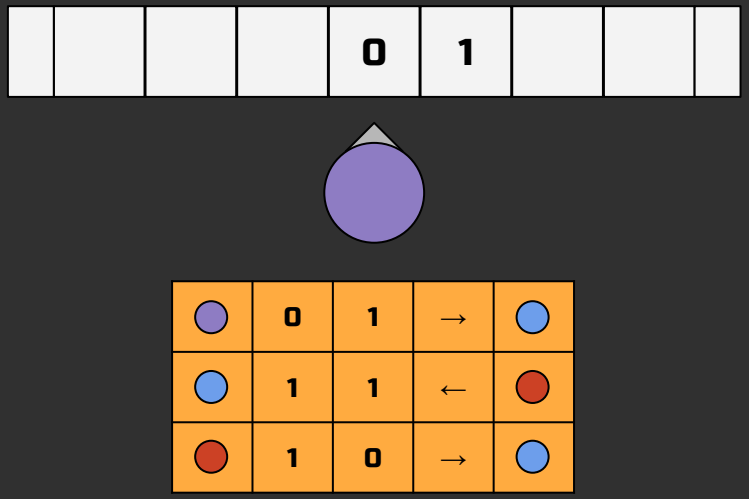
Turing’s Universal Machine



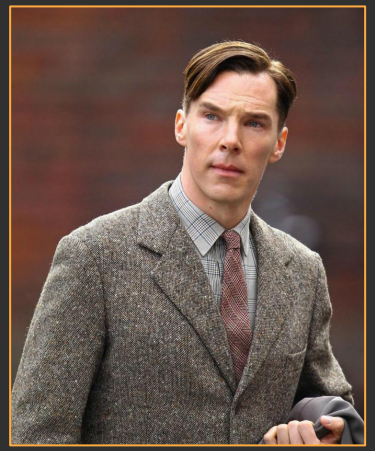


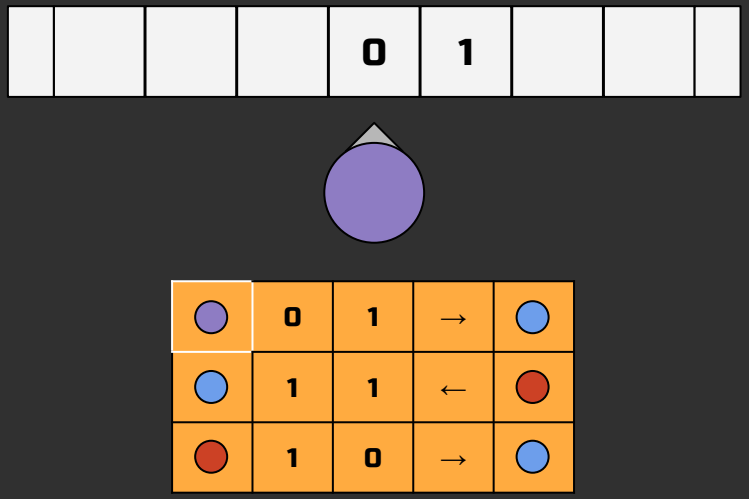
Turing's Universal Machine



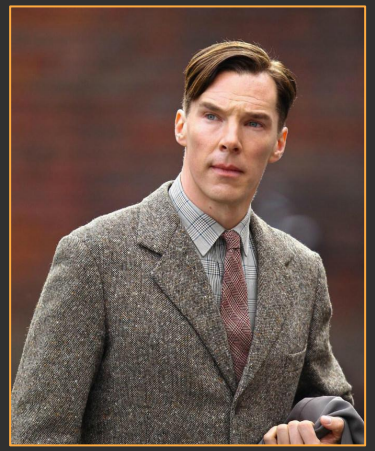


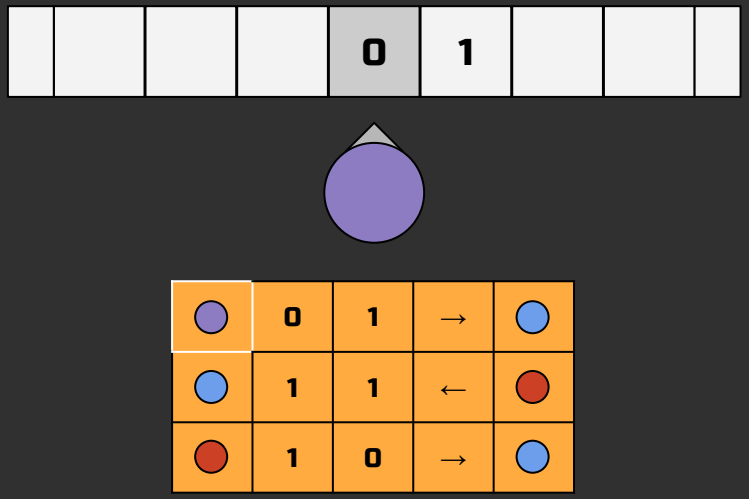
Turing’s Universal Machine



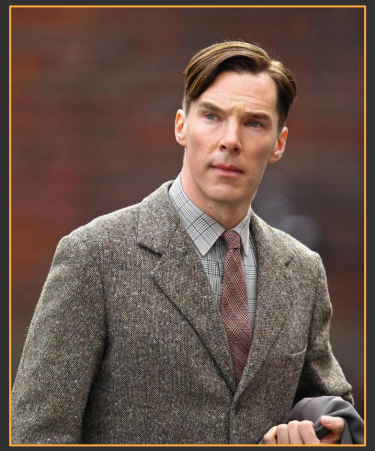


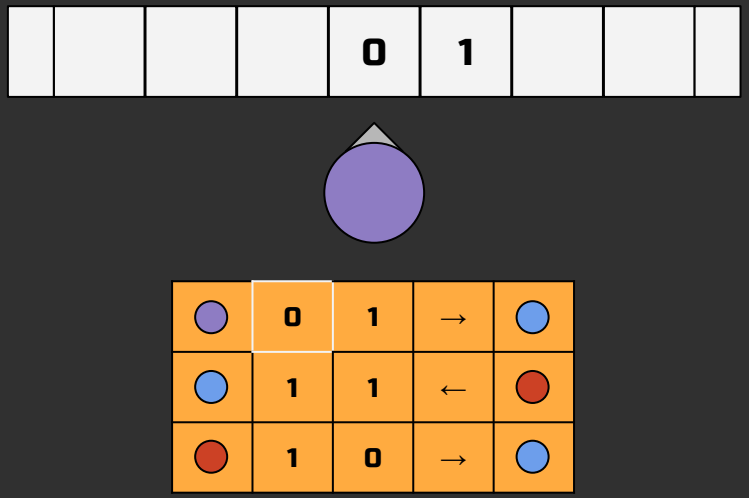
Turing’s Universal Machine



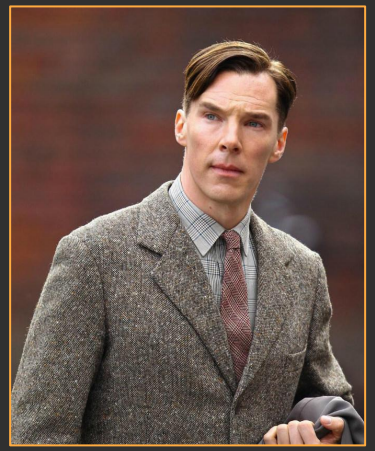


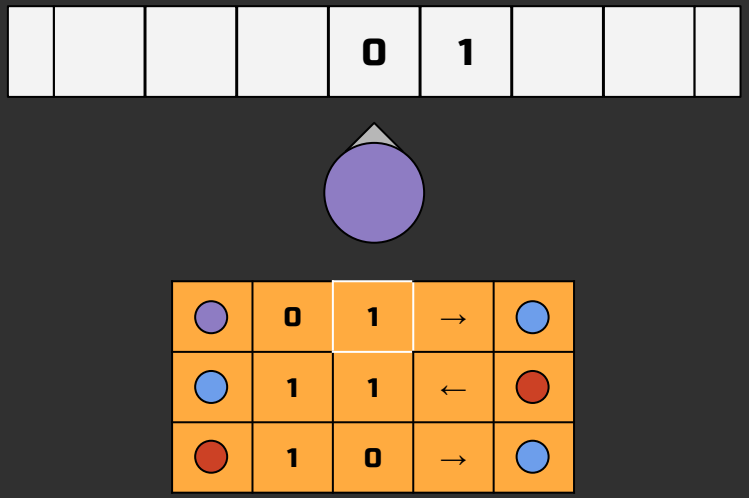
Turing’s Universal Machine



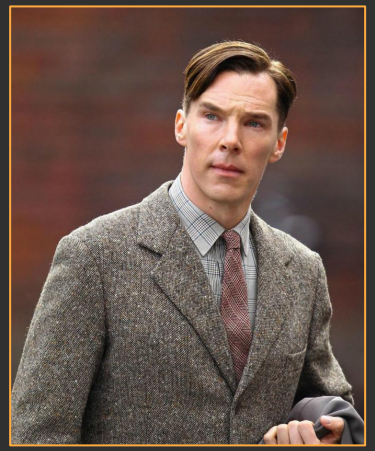


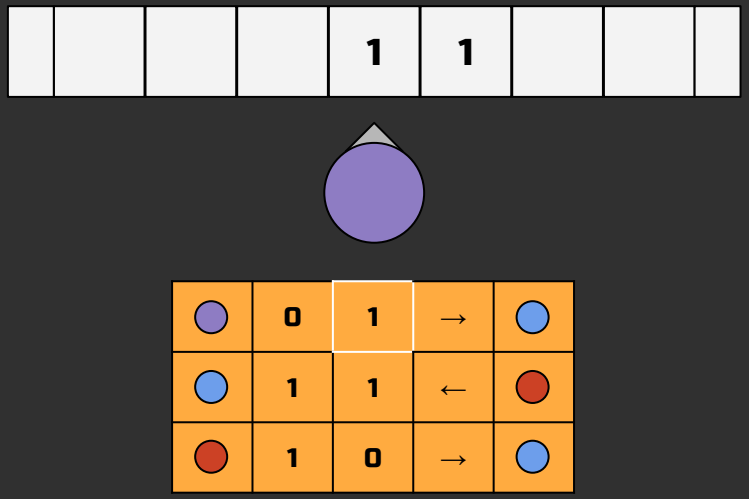
Turing’s Universal Machine



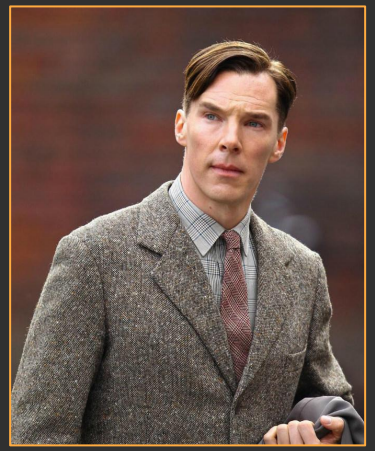


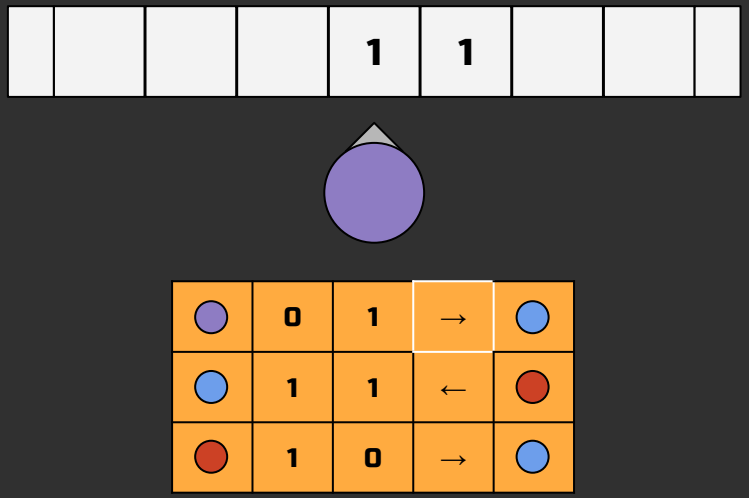
Turing’s Universal Machine



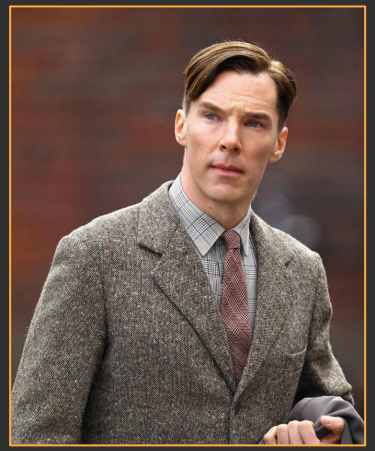


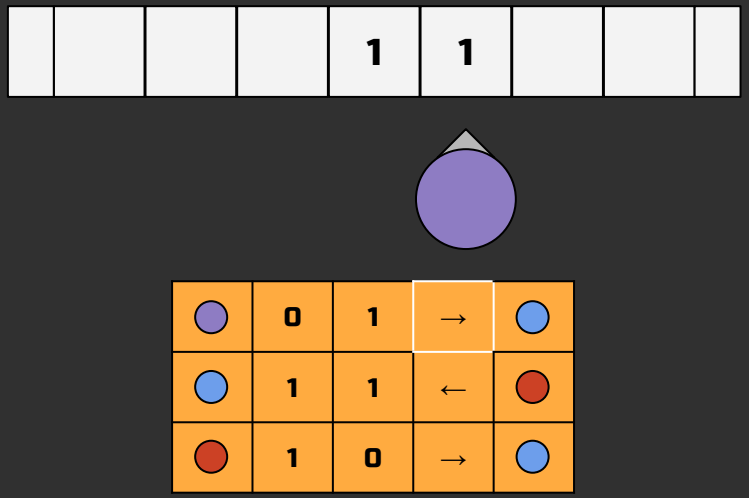
Turing’s Universal Machine



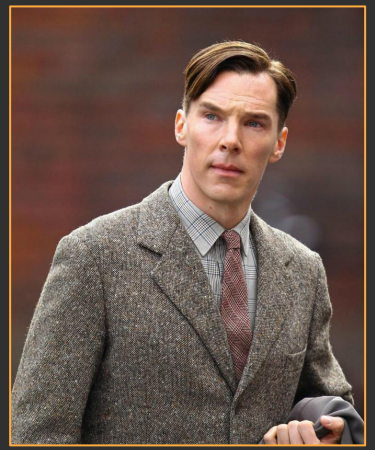


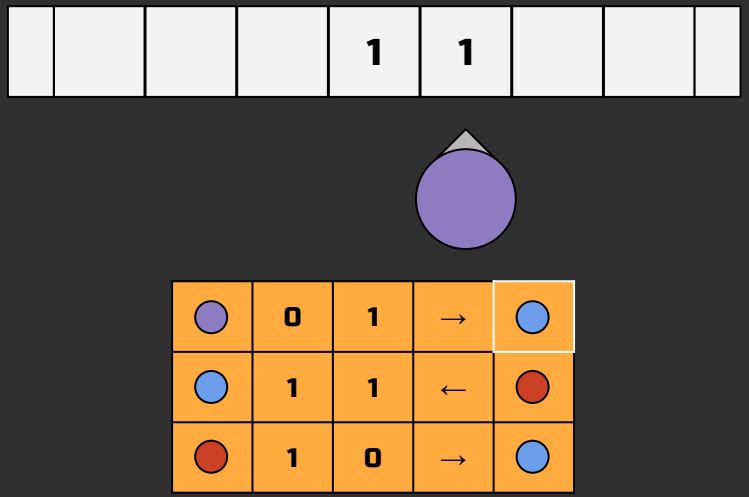
Turing’s Universal Machine



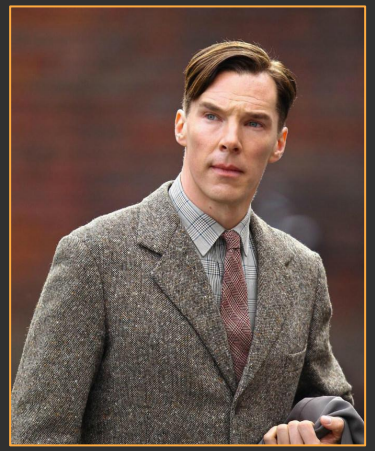


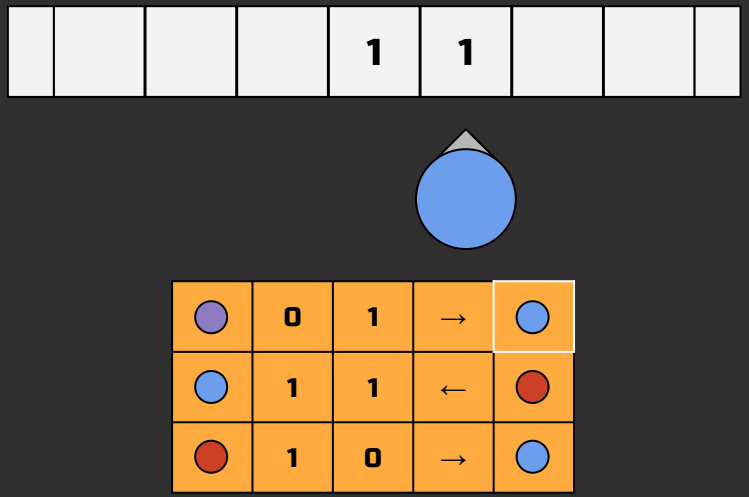
Turing’s Universal Machine



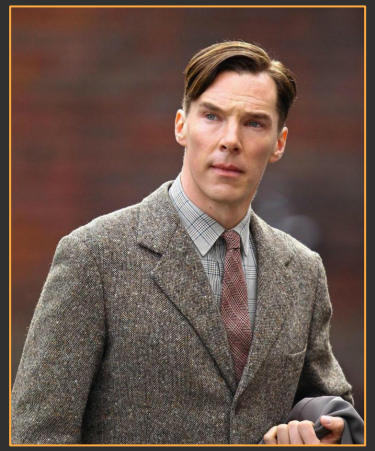


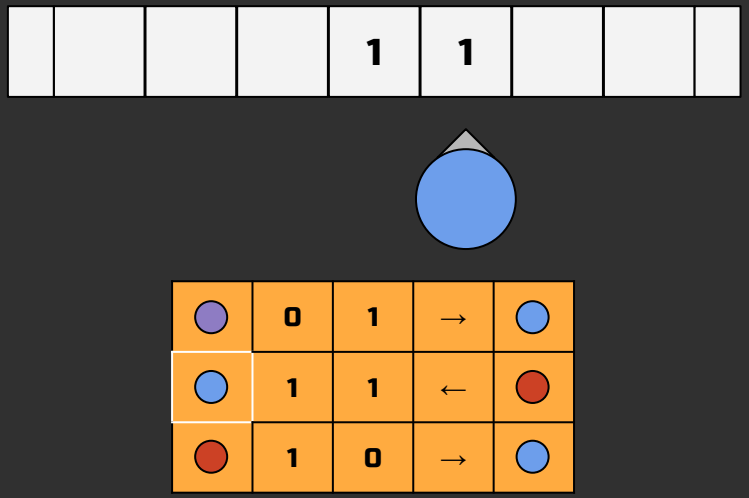
Turing’s Universal Machine



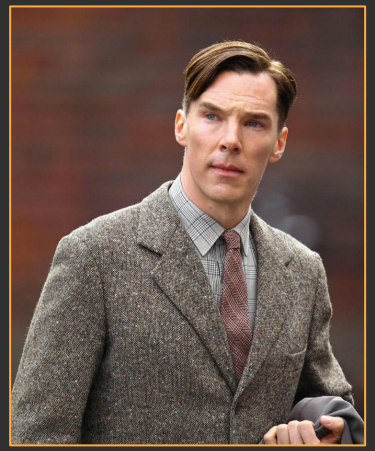


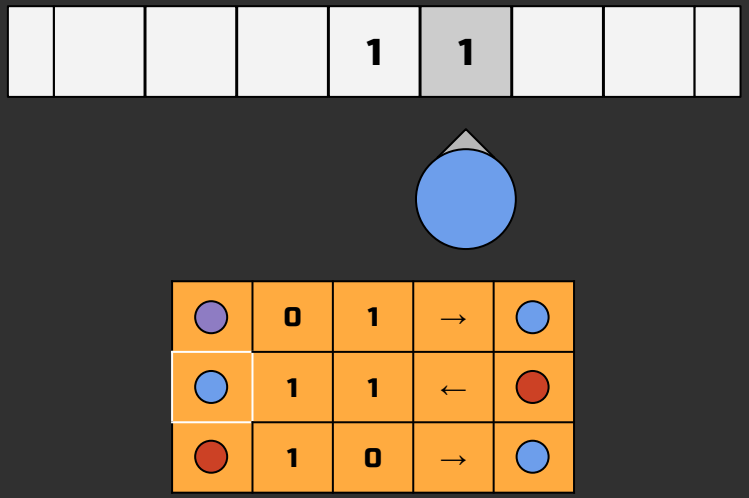
Turing’s Universal Machine



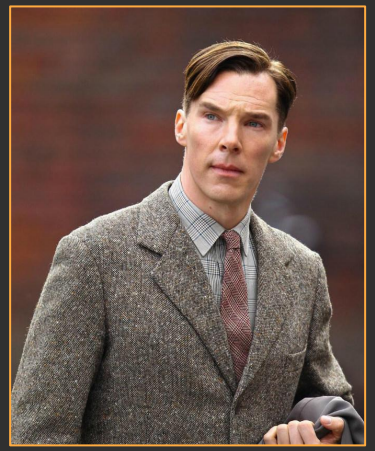


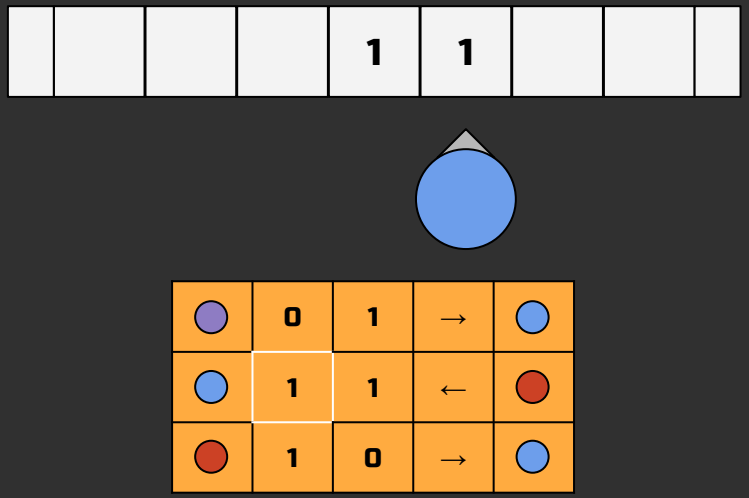
Turing’s Universal Machine



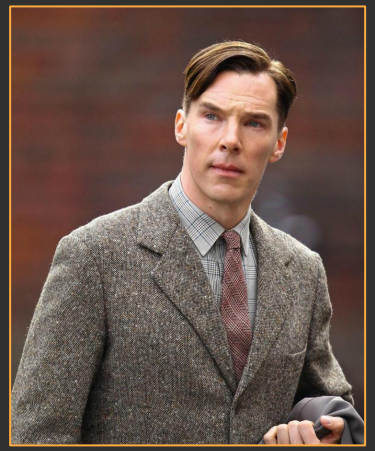


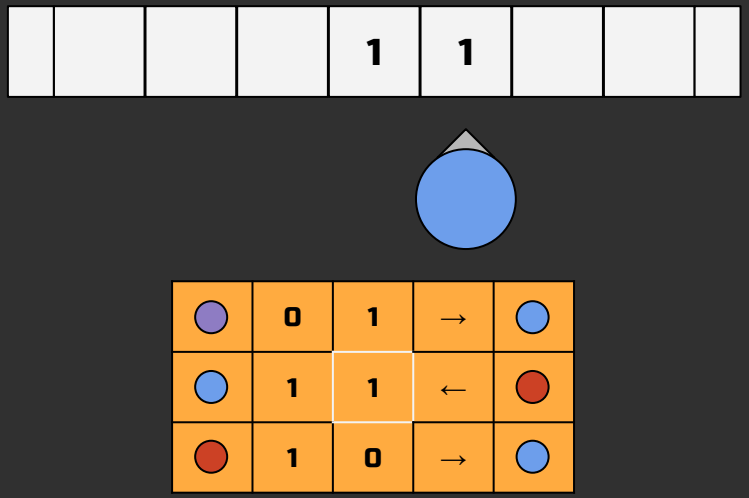
Turing’s Universal Machine



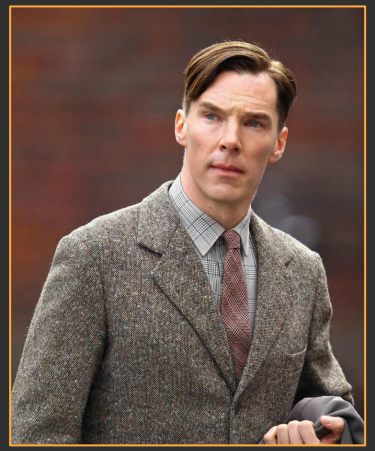


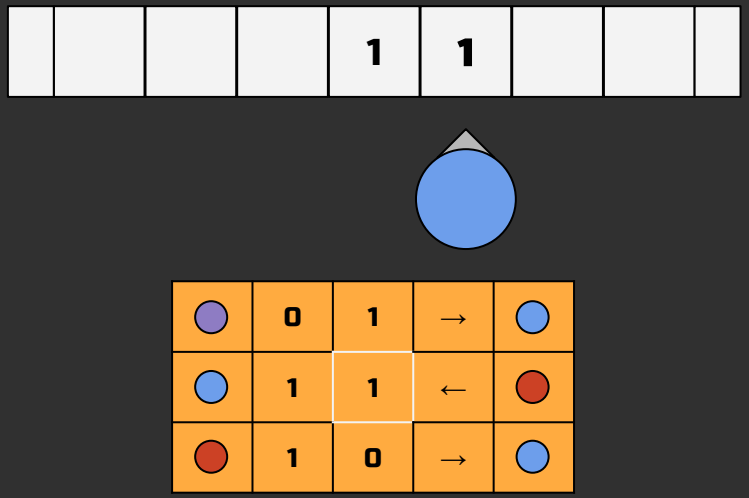
Turing’s Universal Machine



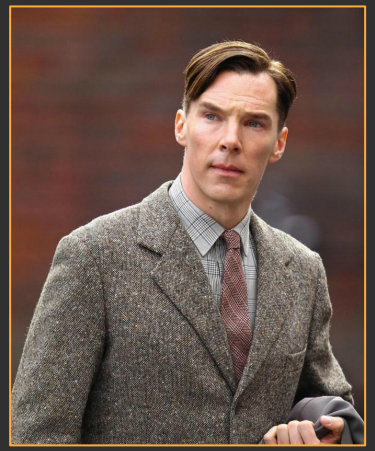


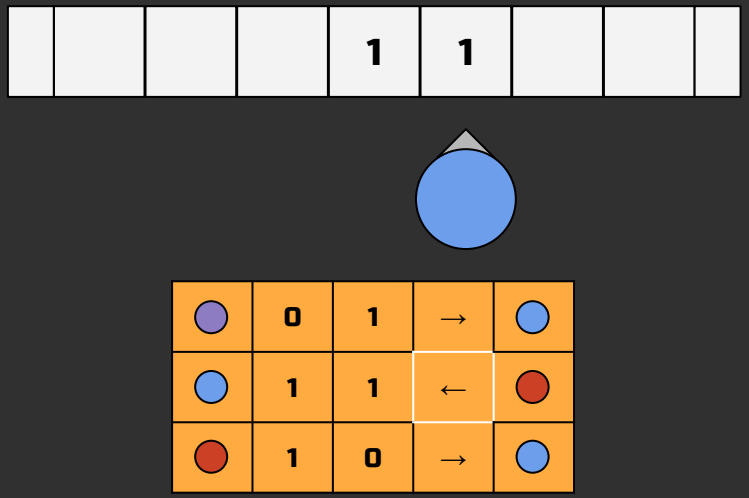
Turing’s Universal Machine



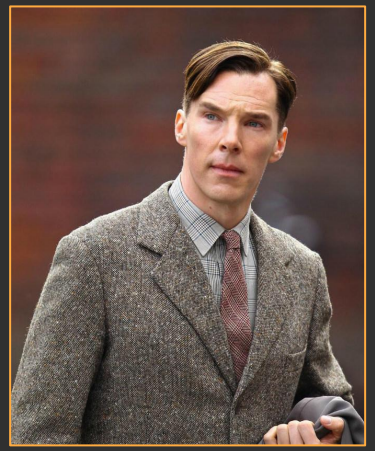


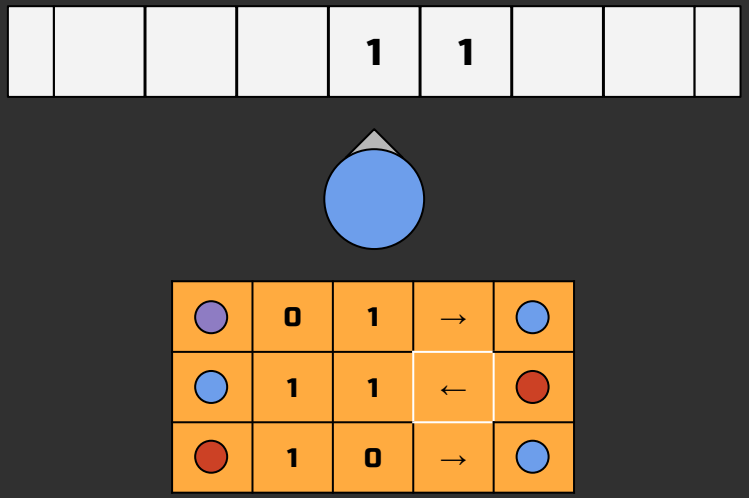
Turing’s Universal Machine



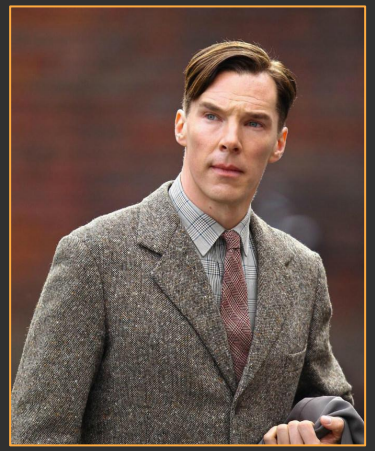


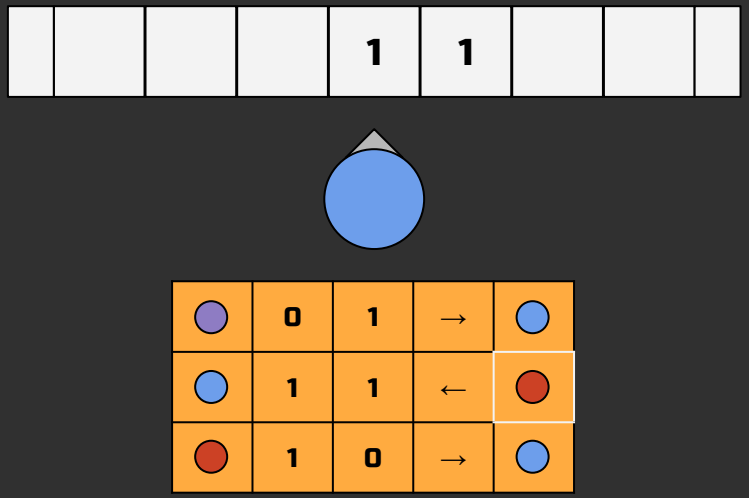
Turing’s Universal Machine



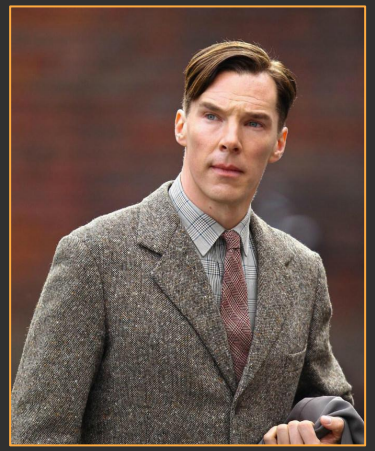


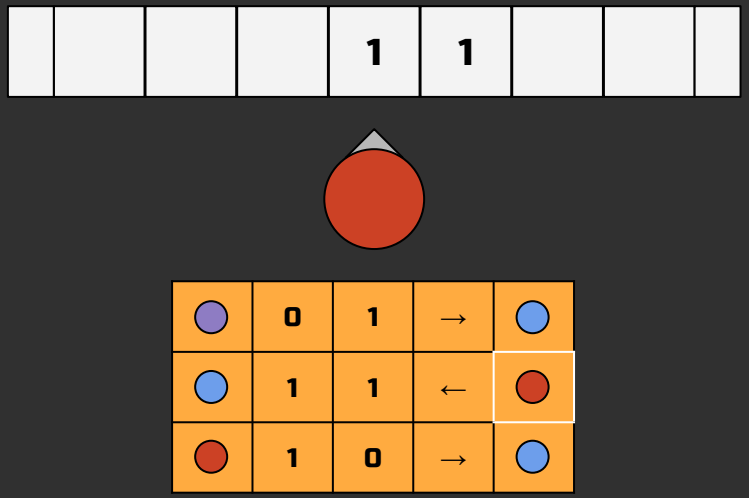
Turing’s Universal Machine



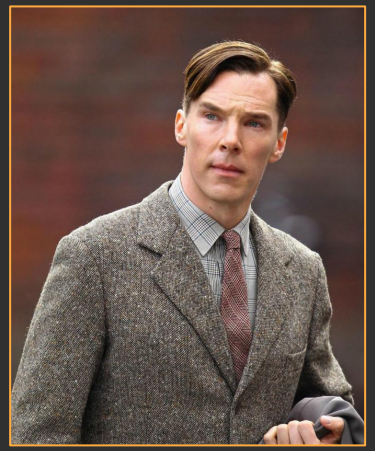


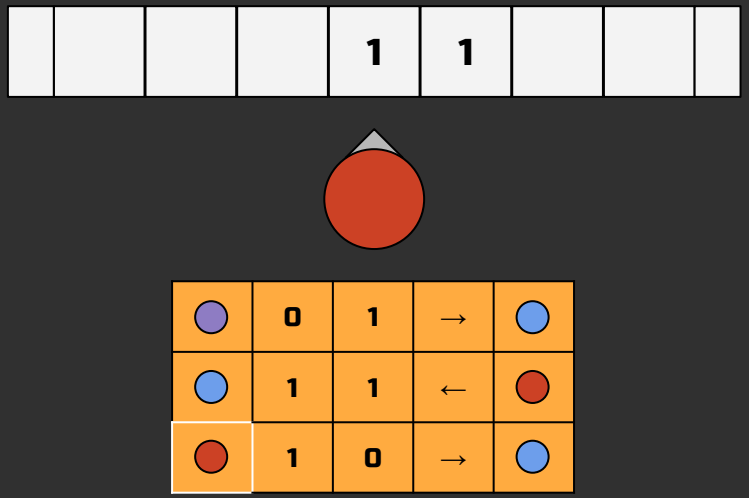
Turing’s Universal Machine



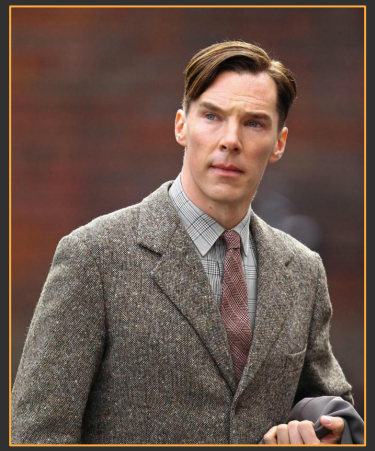


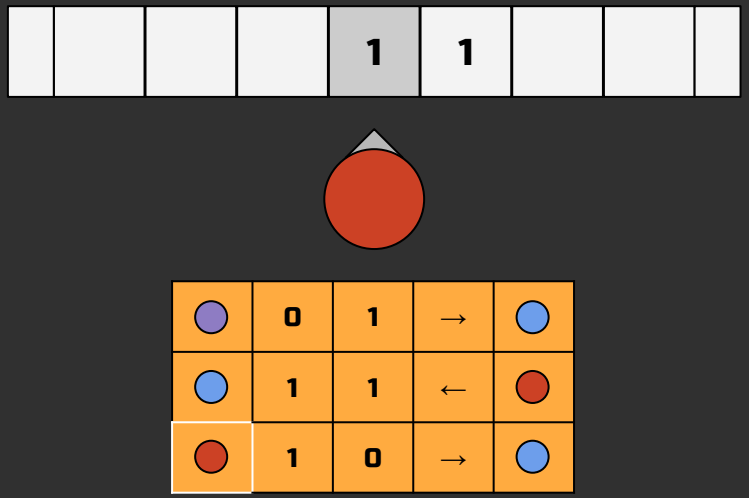
Turing’s Universal Machine



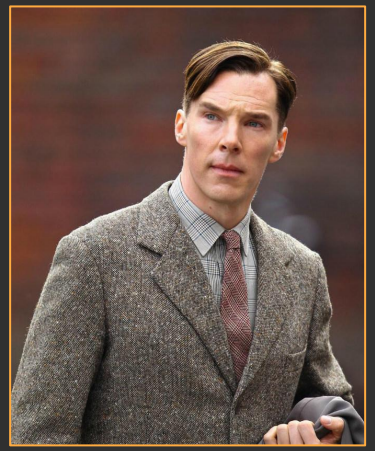


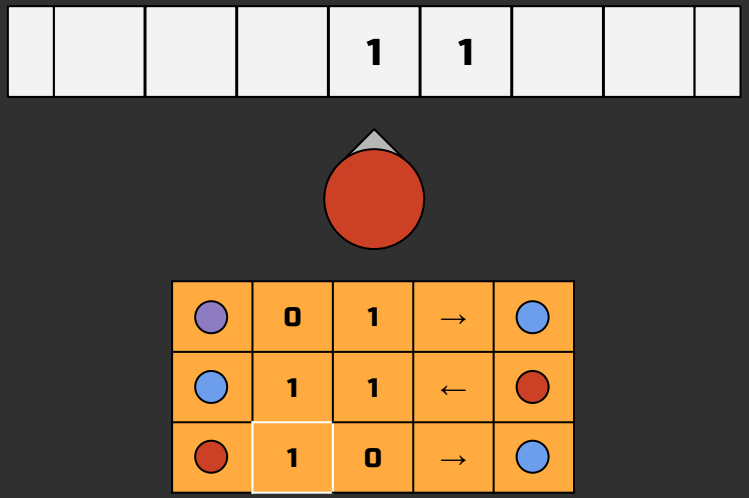
Turing’s Universal Machine



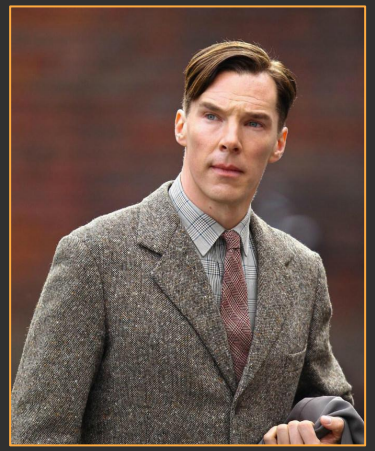


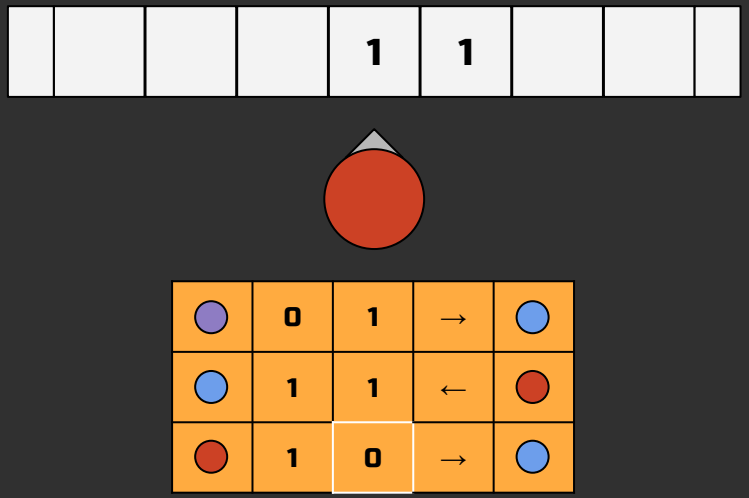
Turing’s Universal Machine



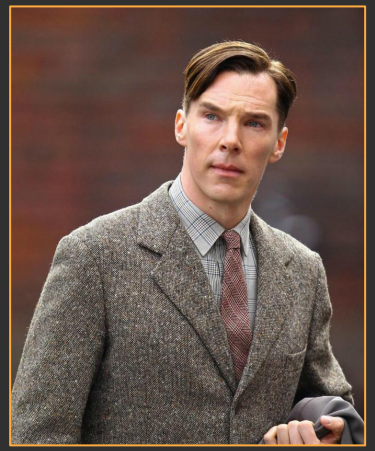


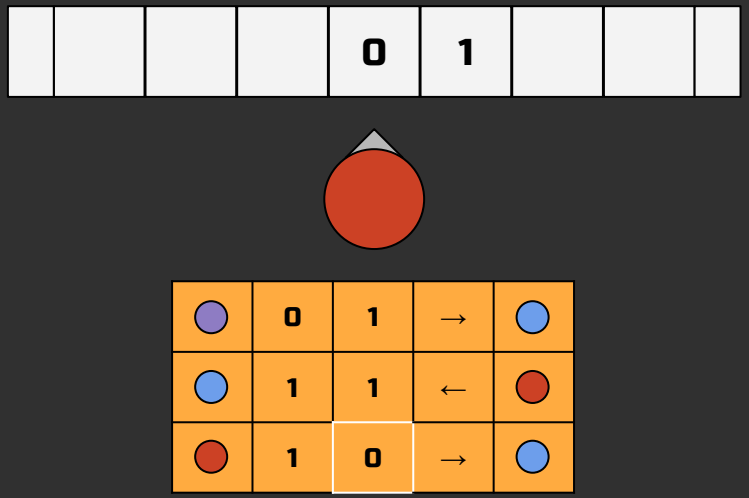
Turing’s Universal Machine



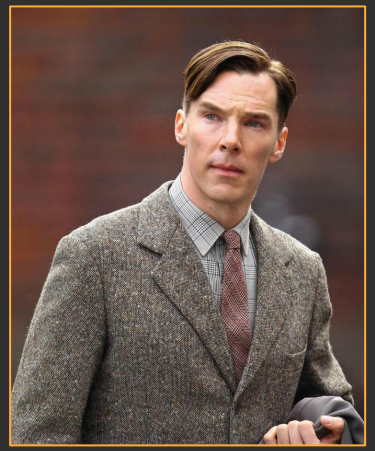


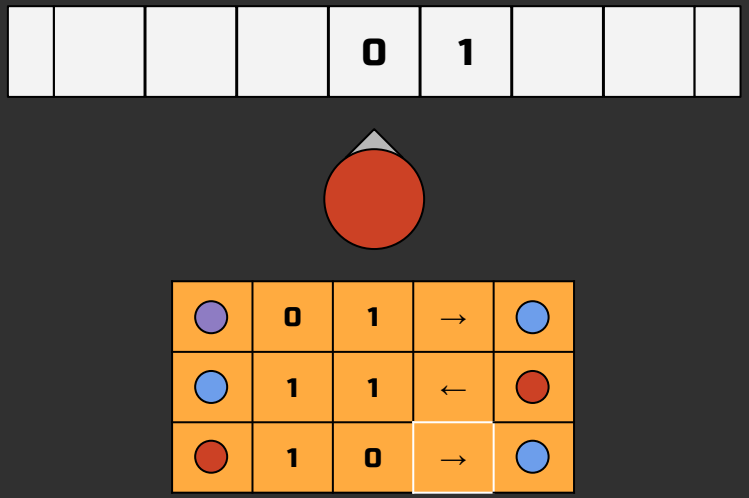
Turing’s Universal Machine



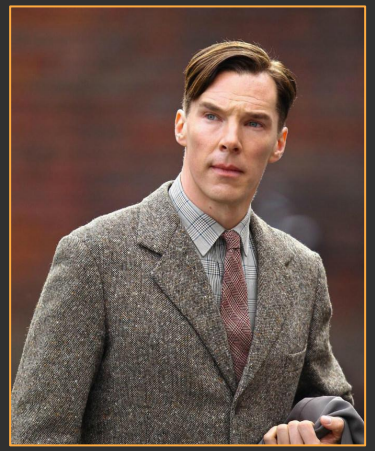


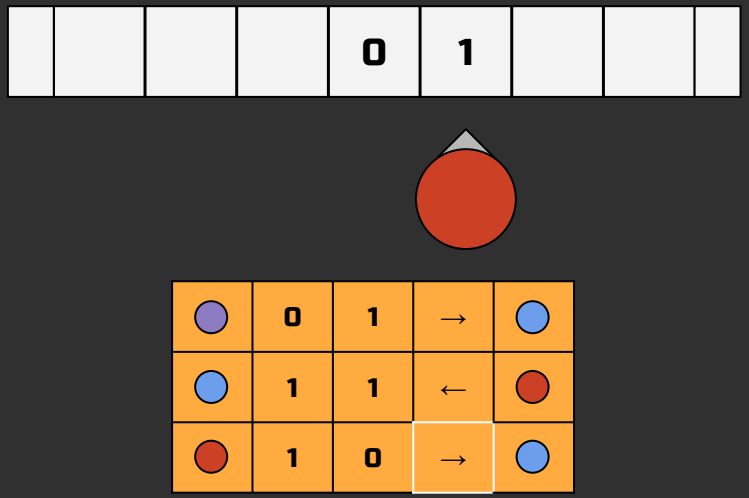
Turing’s Universal Machine



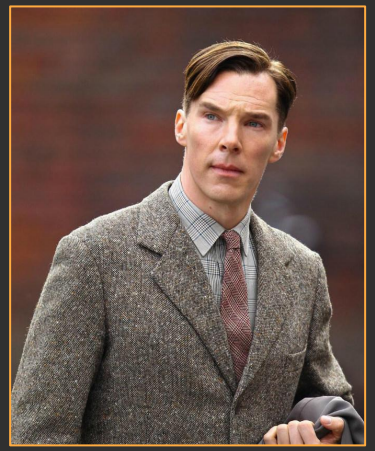


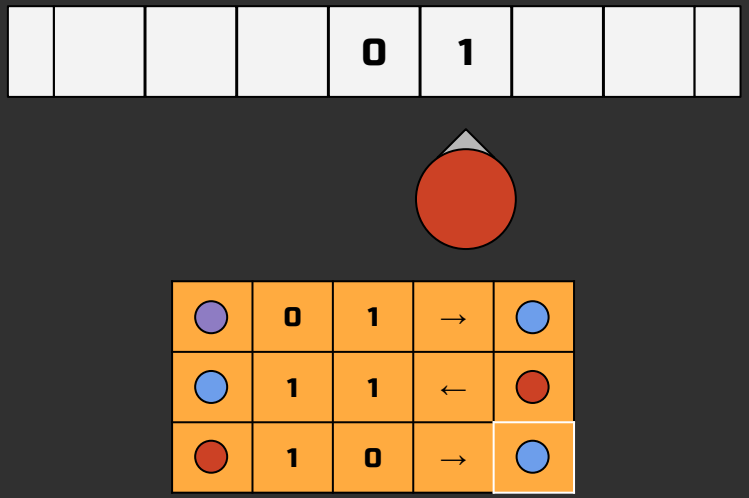
Turing’s Universal Machine



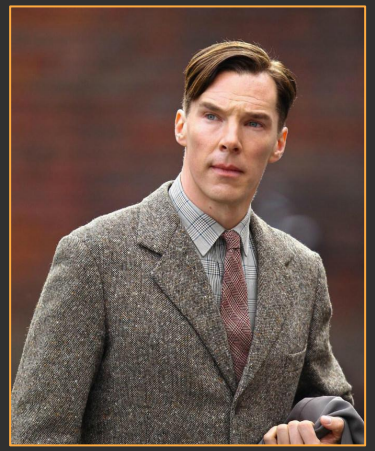


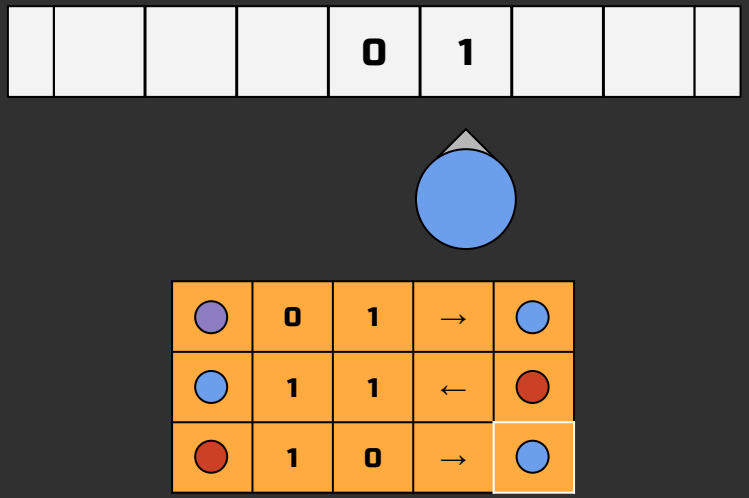
Turing’s Universal Machine



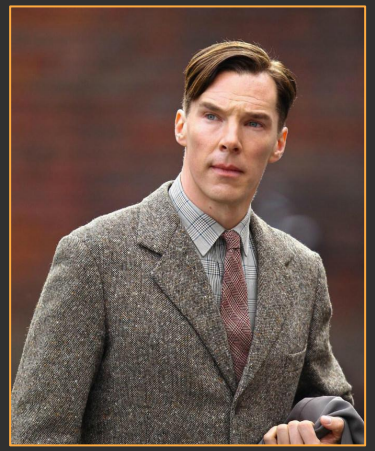


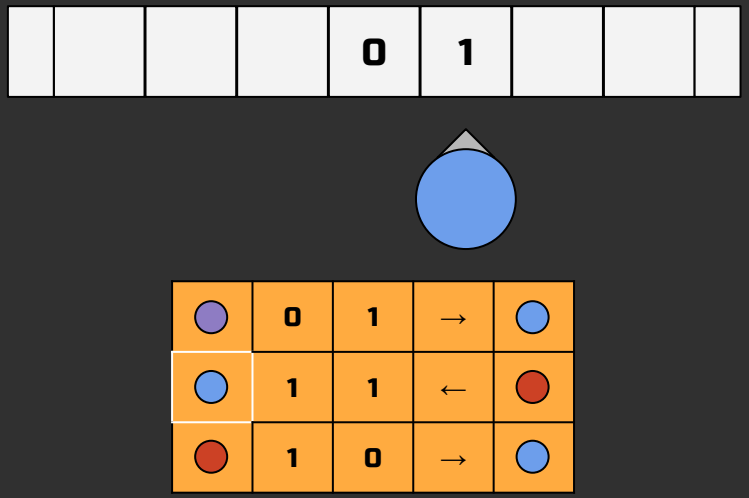
Turing’s Universal Machine



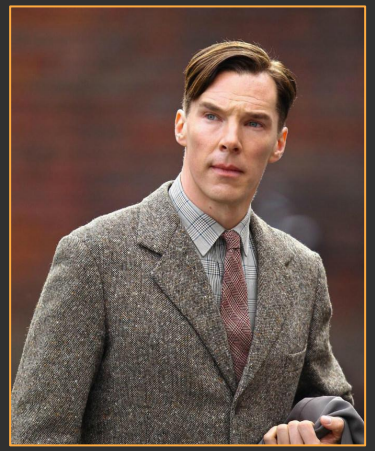


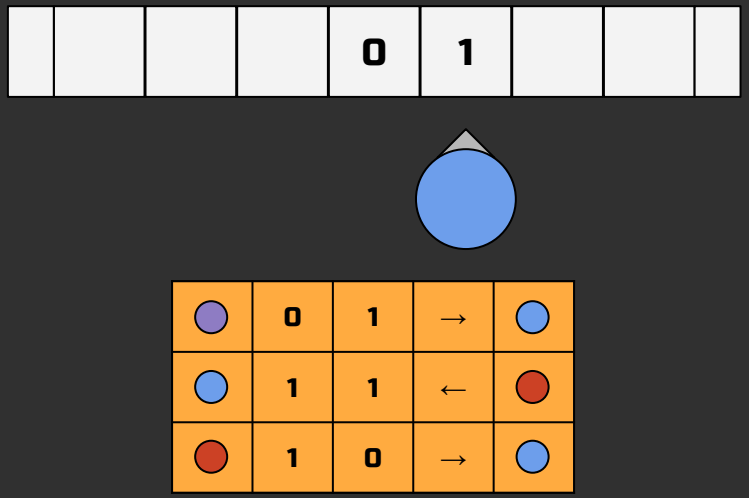
Turing’s Universal Machine





Turing’s Universal Machine





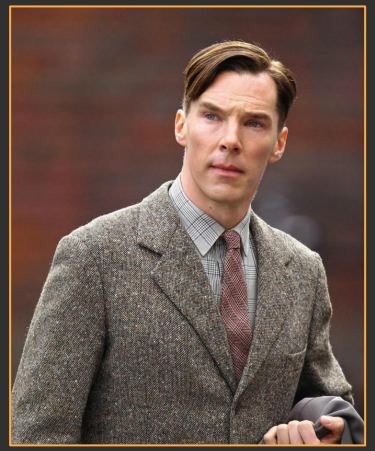
Turing’s Universal Machine

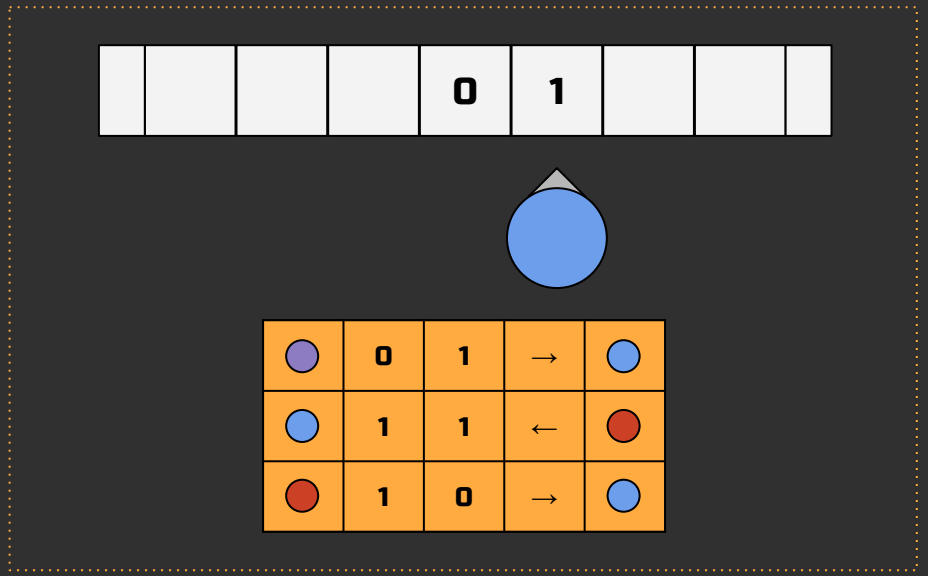
				0	1			

ЗАЧЕММ??

	0	1	→
	1	1	←
	1	0	→

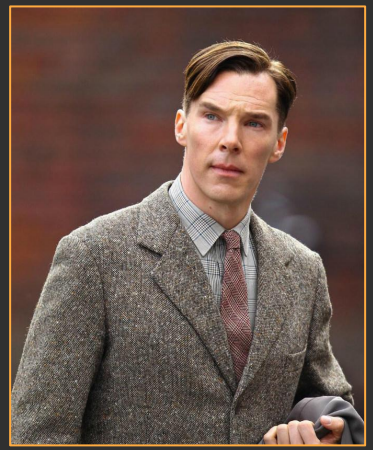
Turing’s Universal Machine

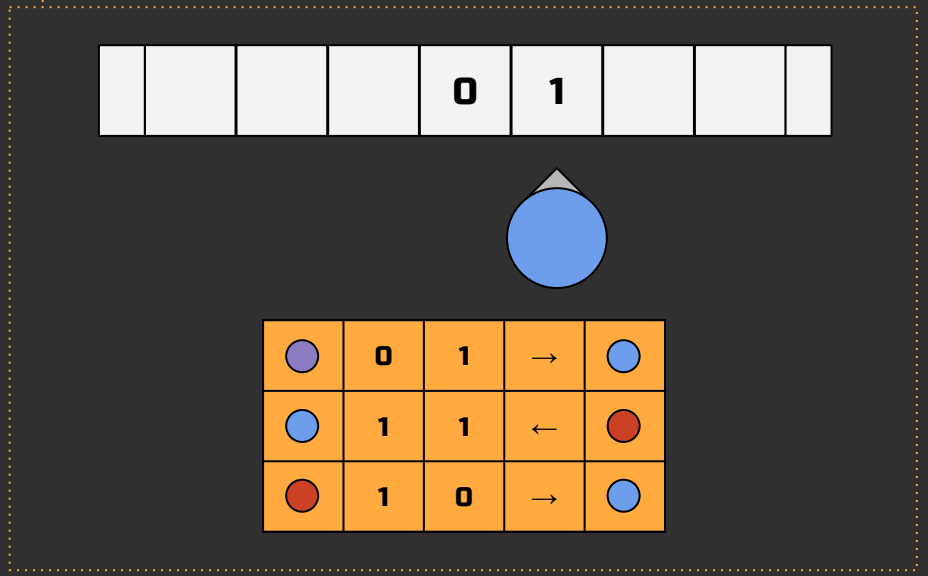




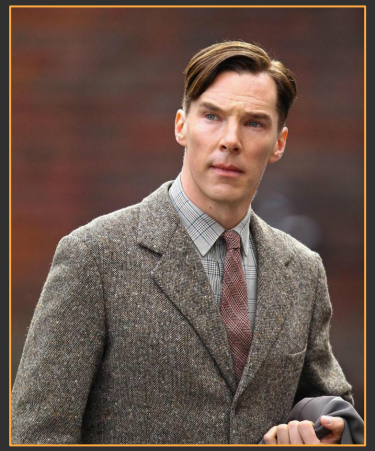
Turing’s Universal Machine

Проще работать



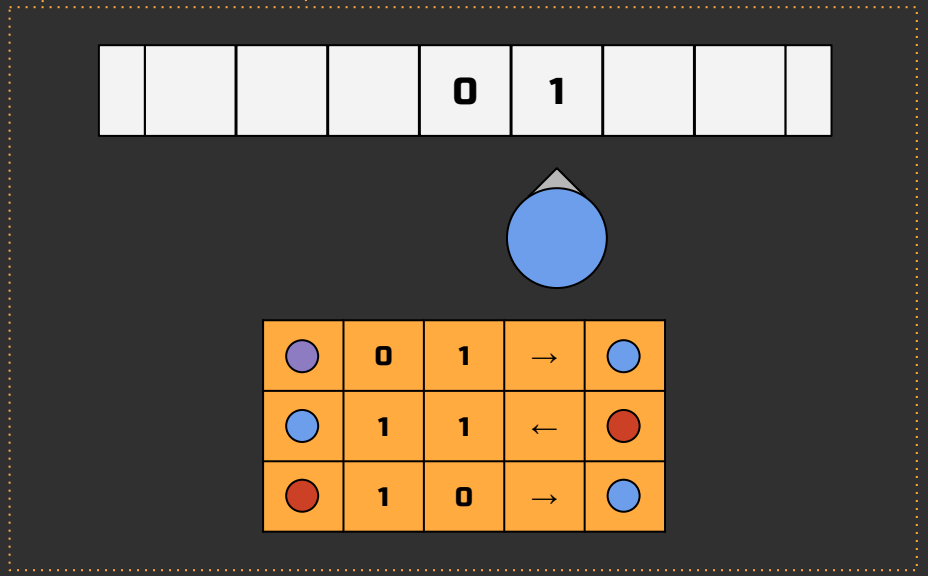


Turing's Universal Machine

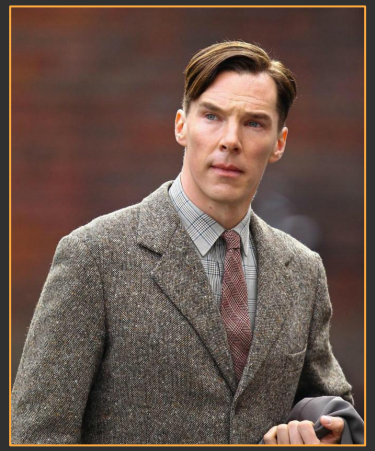


Проще работать

Можно выразить любое вычисление

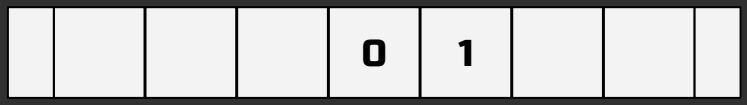


Turing's Universal Machine



Проще работать

Можно выразить любое вычисление → Доказательства обобщаются



	0	1	→	
	1	1	←	
	1	0	→	

Turing's Universal Machine

```
F, A, B ::= x
          | (λx.B)
          | (F A)
```

```
F, A, B ::= x
          | (λx.B)
          | (F A)
```

λ-calculus

```
F, A, B ::= x
          | (λx.B)
          | (F A)
```

λ-calculus

```
F, A, B ::= x
          | (λx.B)
          | (F A)
```

Имя параметра (или просто имя)

λ-calculus

```
F, A, B ::= x
          | (λx.B)
          | (F A)
```

Имя параметра (или просто имя)

Анонимная функция

λ-calculus

```
F, A, B ::= x
          | x => B
          | (F A)
```

Имя параметра (или просто имя)

Анонимная функция

λ-calculus

```
F, A, B ::= x
          | x => B
          | (F A)
```

Имя параметра (или просто имя)

Анонимная функция

Вызов функции

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Имя параметра (или просто имя)

Анонимная функция

Вызов функции

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Имя параметра (или просто имя)

Анонимная функция

Вызов функции

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Имя параметра (или просто имя)

Анонимная функция

Вызов функции

λ-calculus


```
F, A, B ::= x
          | x => B
          | F(A)
```

Имя параметра (или просто имя)

Анонимная функция

Вызов функции

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

```
x => x
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

x => x — *Identity*

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

```
f => f(f)
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

`f => f(f)` — *Self application*

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
f => f(f)
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))(f => f(f))
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))(f => f(f))
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))(
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))(                )
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))
```

```
(
)
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))(f => f(f))
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))(f => f(f))

(f => f(f))(                )
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))(f => f(f))

(f => f(f))((f => f(f)))
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))(f => f(f))

(f => f(f))(f => f(f))
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

```
(f => f(f))(f => f(f))

(f => f(f))(f => f(f)) — Recursion
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

- **Identity**
- **Self application**
- **Recursion**

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

- **Identity**
- **Self application**
- **Recursion**

```
(x, y) => x
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

- **Identity**
- **Self application**
- **Recursion**

```
(x, y) => x — TRUE
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

- **Identity**
- **Self application**
- **Recursion**

```
(x, y) => x — TRUE
(x, y) => y
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

Identity

Self application

Recursion

```
(x, y) => x — TRUE
(x, y) => y — FALSE
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

- **Identity**
- **Self application**
- **Recursion**

```
(x, y) => x — TRUE
(x, y) => y — FALSE
(p, q) => p(q, p)
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

- **Identity**
- **Self application**
- **Recursion**

```
(x, y) => x — TRUE
(x, y) => y — FALSE
(p, q) => p(q, p) — AND
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

- **Identity**
- **Self application**
- **Recursion**

```
(x, y) => x — TRUE
(x, y) => y — FALSE
(p, q) => p(q, p) — AND
(p, q) => p(p, q) — OR
```

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

- Identity
- Boolean Logic
- Self application
- Recursion

λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

- Identity
- Boolean Logic
- Self application
- If-Then-Else
- Recursion

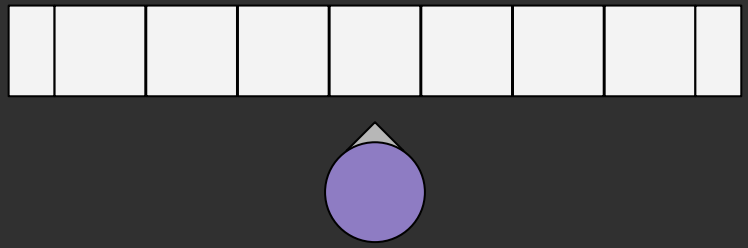
λ-calculus

```
F, A, B ::= x
          | x => B
          | F(A)
```

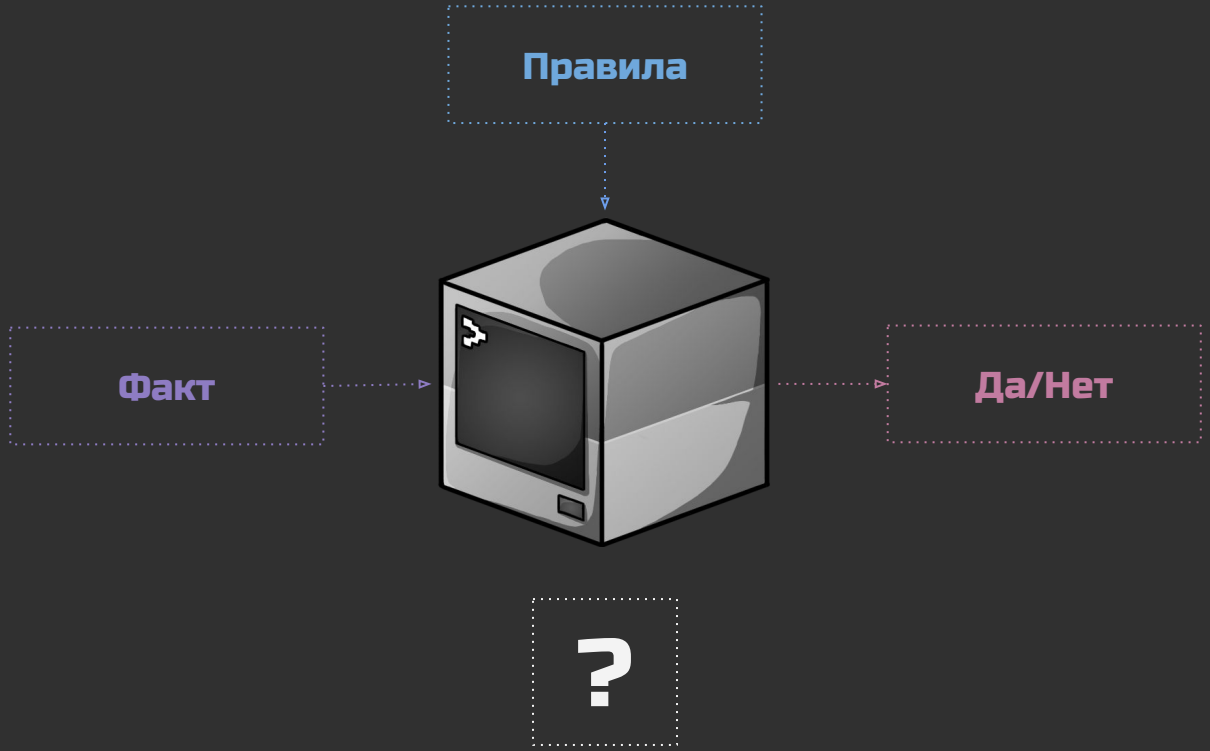
Identity	**Boolean Logic**
Self application	**If-Then-Else**
Recursion	**Cycles**

λ-calculus

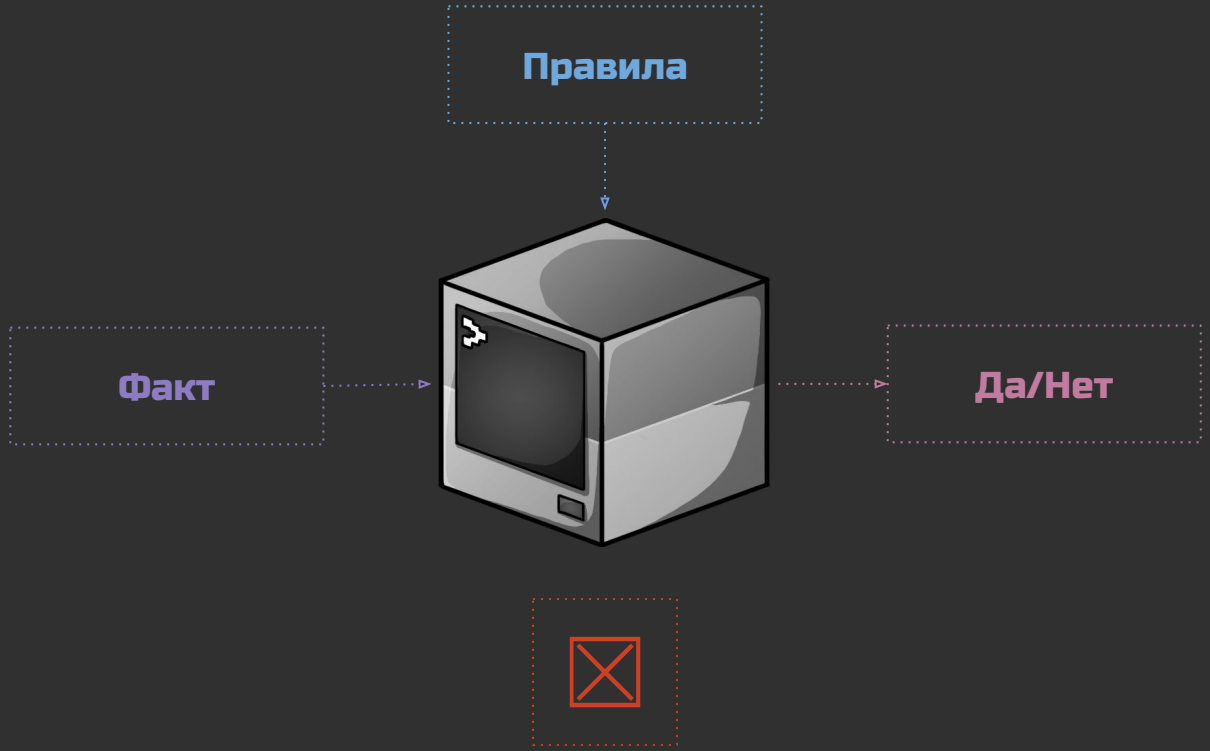
```
F, A, B ::= x
          | (λx.B)
          | (F A)
```



q_1	x_0	a_0	m_0	q_m
q_1	x_1	a_1	m_1	q_n
...	...	...	...	...



Entscheidungsproblem (decision problem)



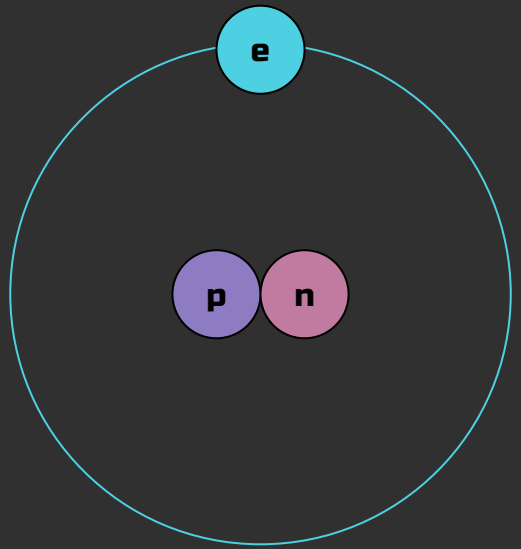
Entscheidungsproblem (decision problem)

Идеи и принципы

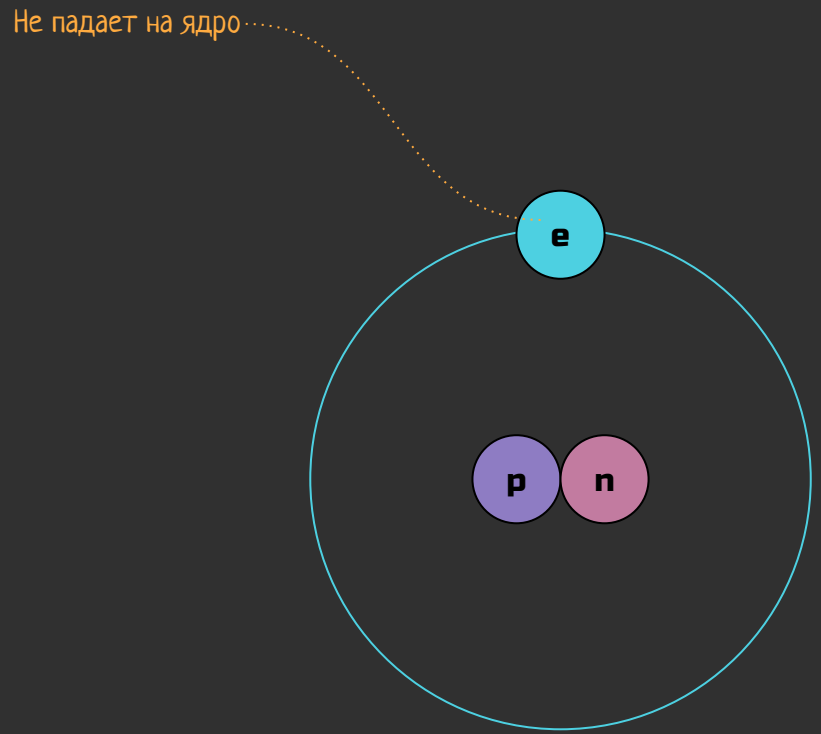
High-order functions

High-order functions

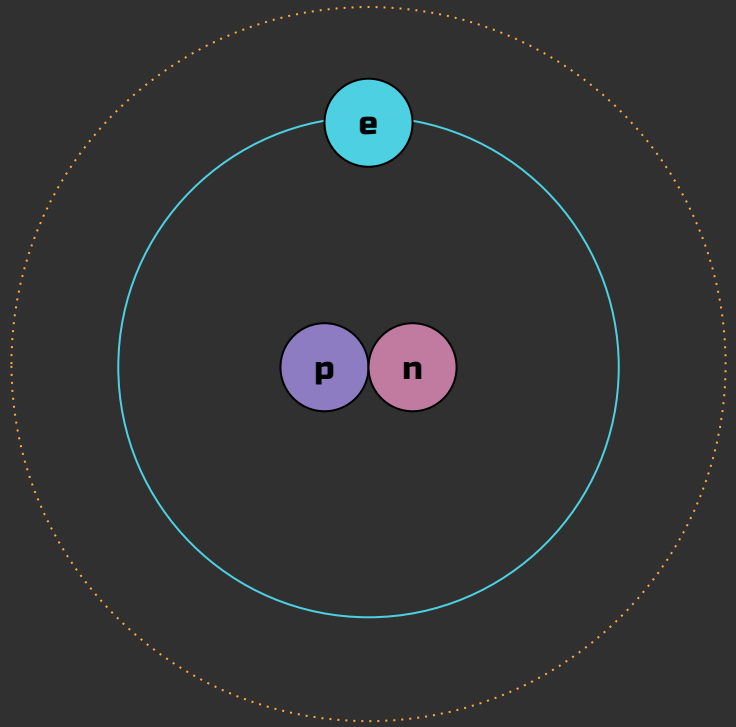
High-order functions



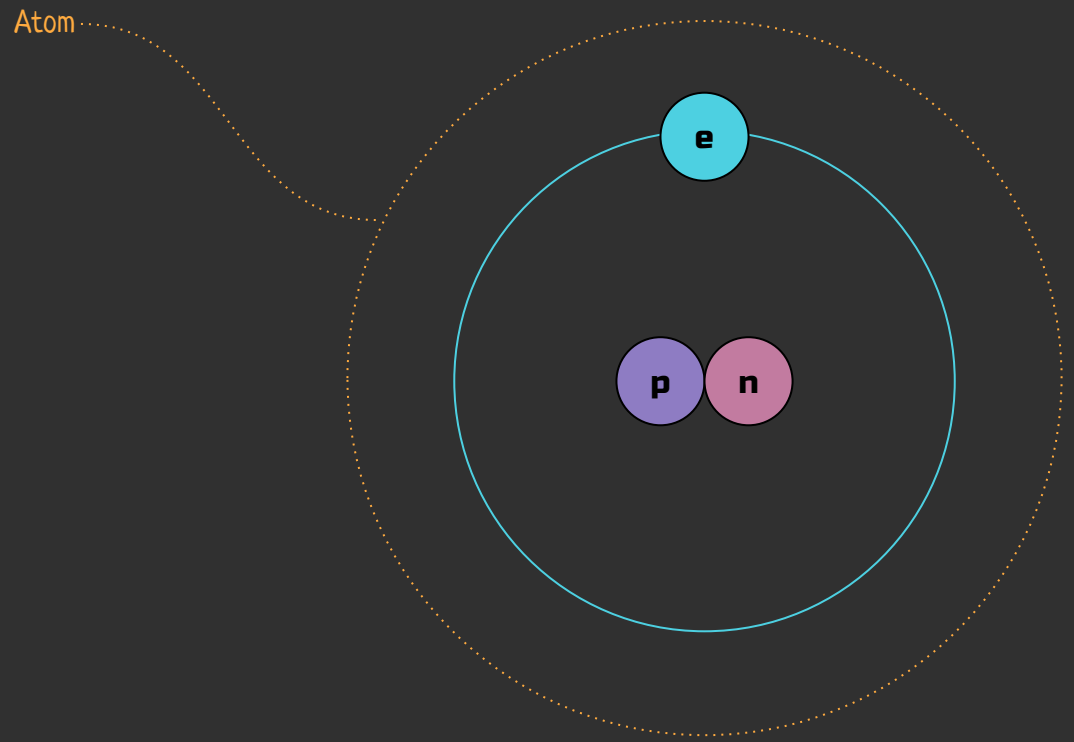
High-order functions



High-order functions

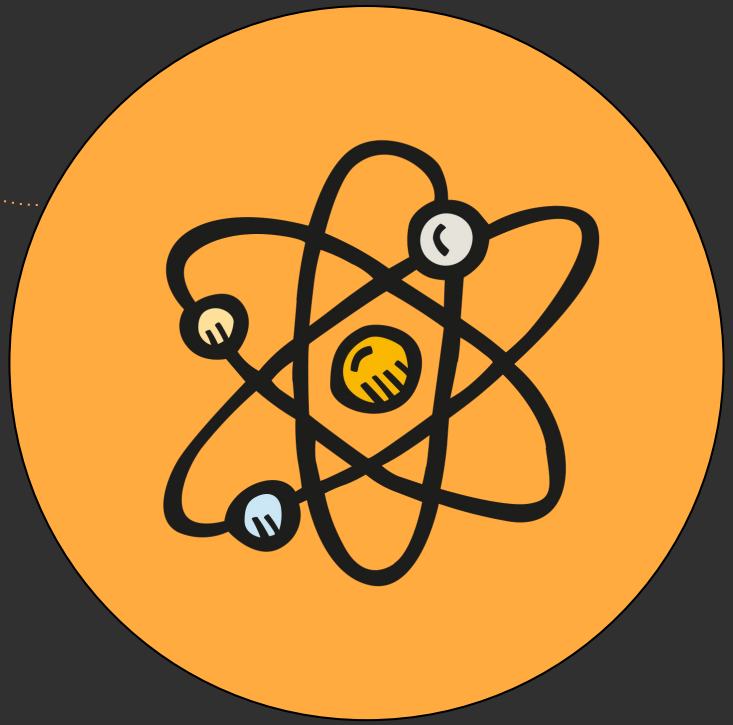


High-order functions



High-order functions

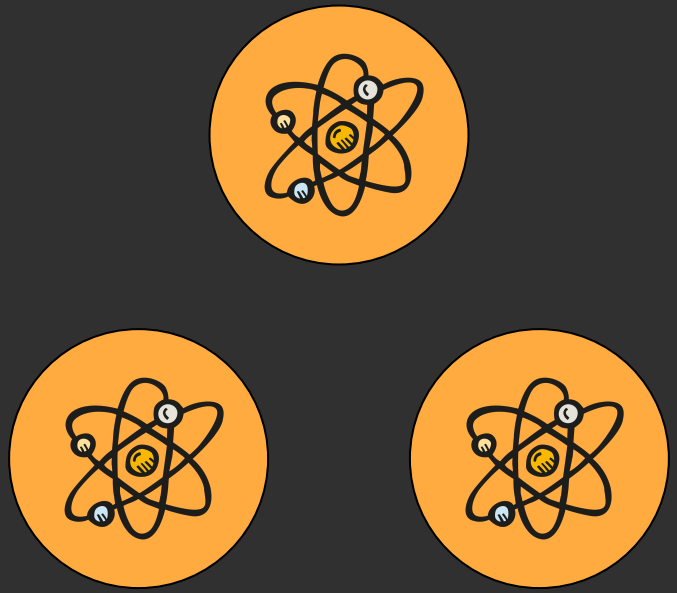
Atom



High-order functions

High-order functions

High-order functions



High-order functions

High-order functions

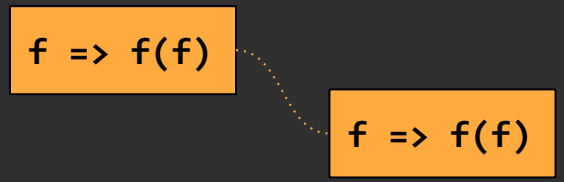
High-order functions

High-order functions

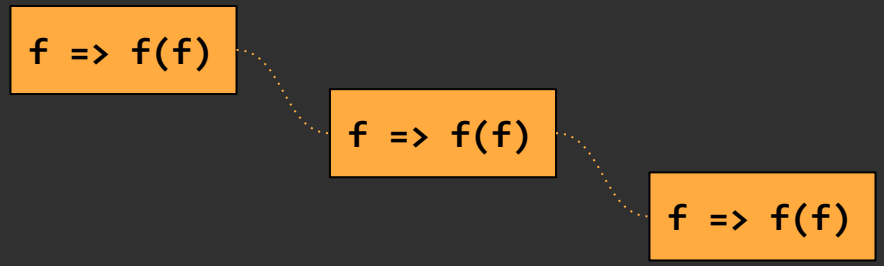
High-order functions

```
f => f(f)
```

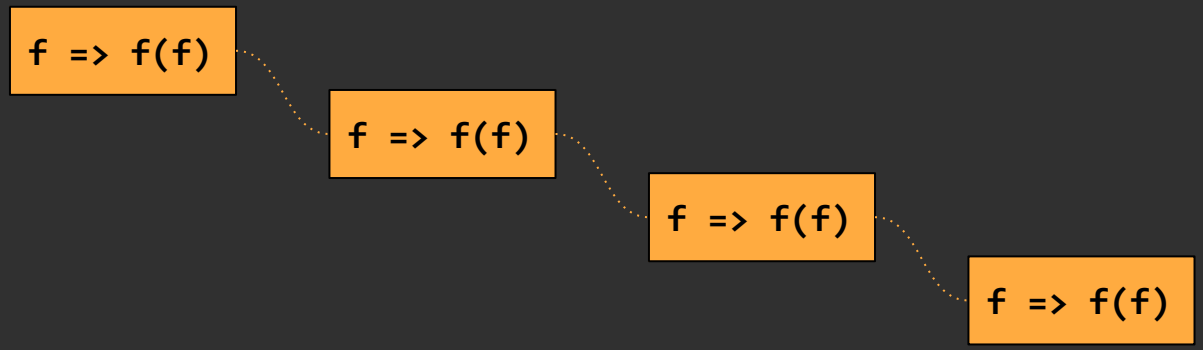
High-order functions



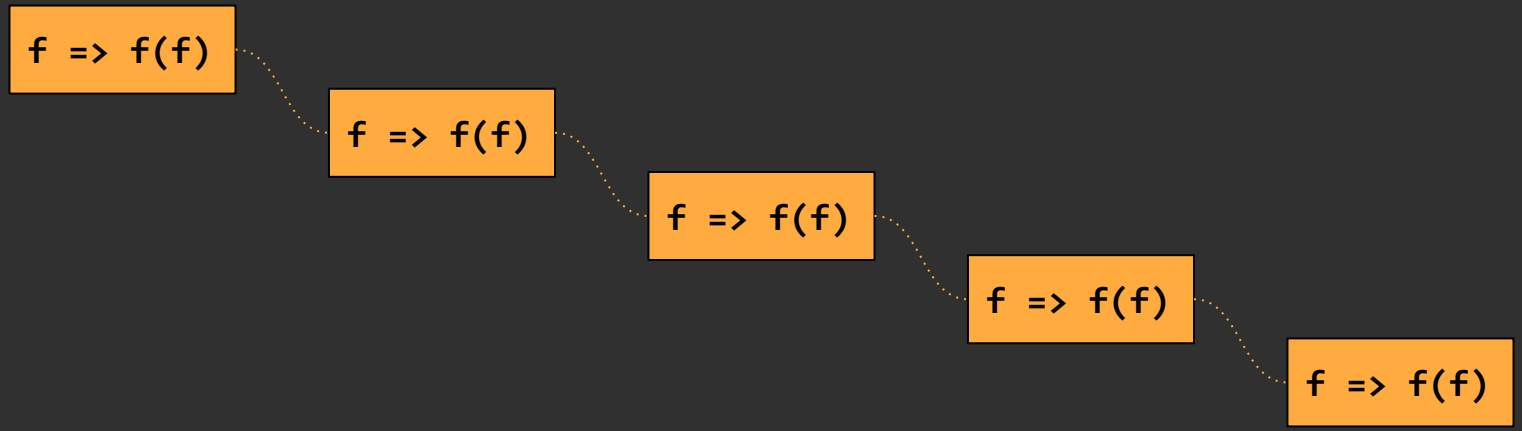
High-order functions



High-order functions



High-order functions

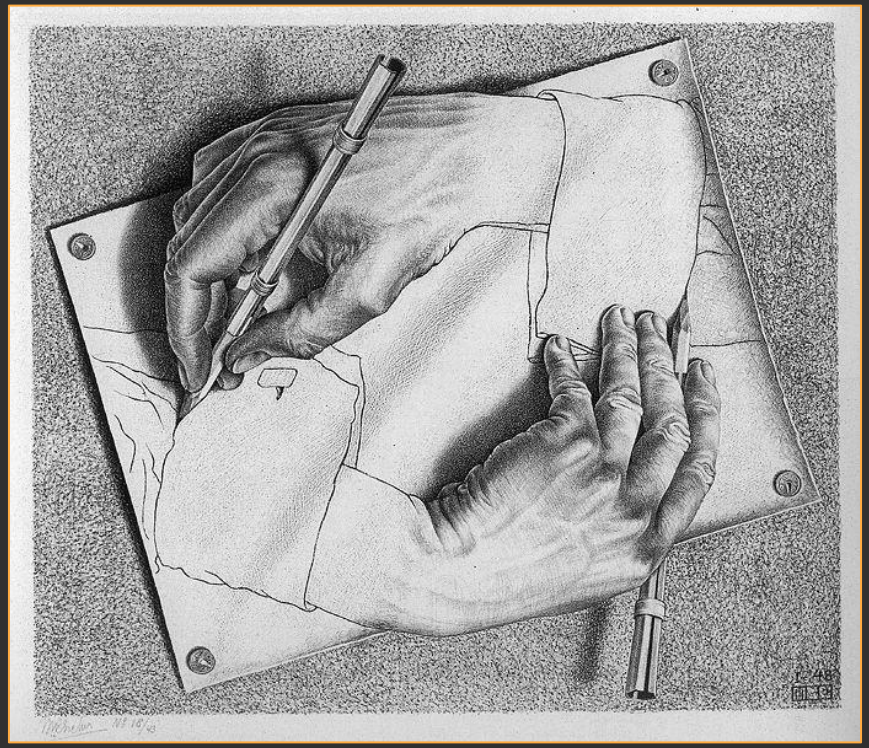


High-order functions

```
private static Func<int, int, int, int, int, int, int> HighOrderGuy(Func<int, Func<int, Func<int, Func<int, Func<int, Func<int, int>>>>>> monster) =>
  (a, b, c, d, e, f) => monster(a)(b)(c)(d)(e)(f);
```

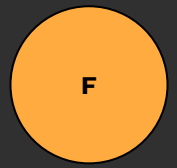
High-order functions

Recursion

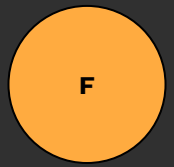


Recursion

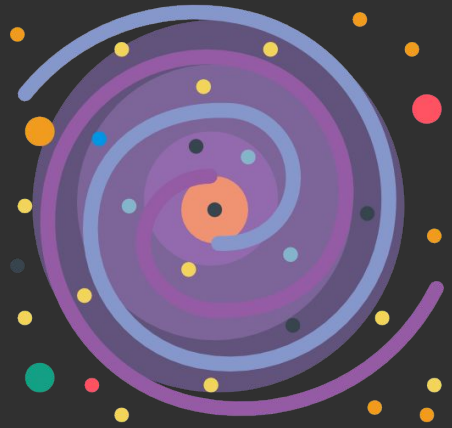
Purity



Purity

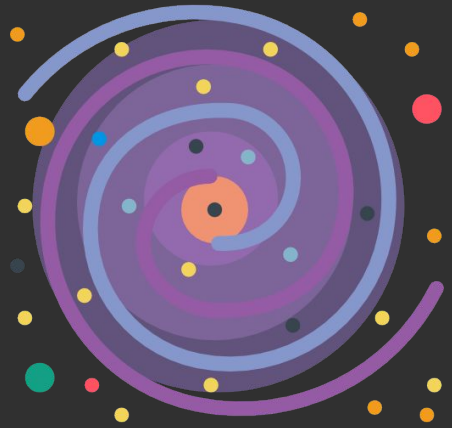


Purity

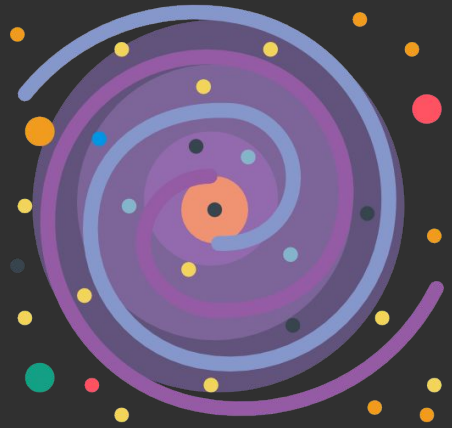


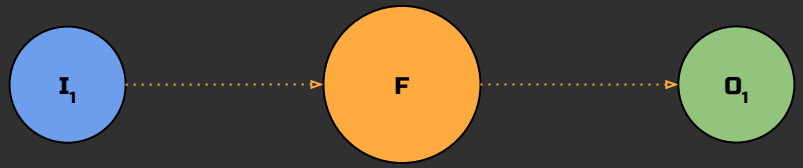
F

Purity

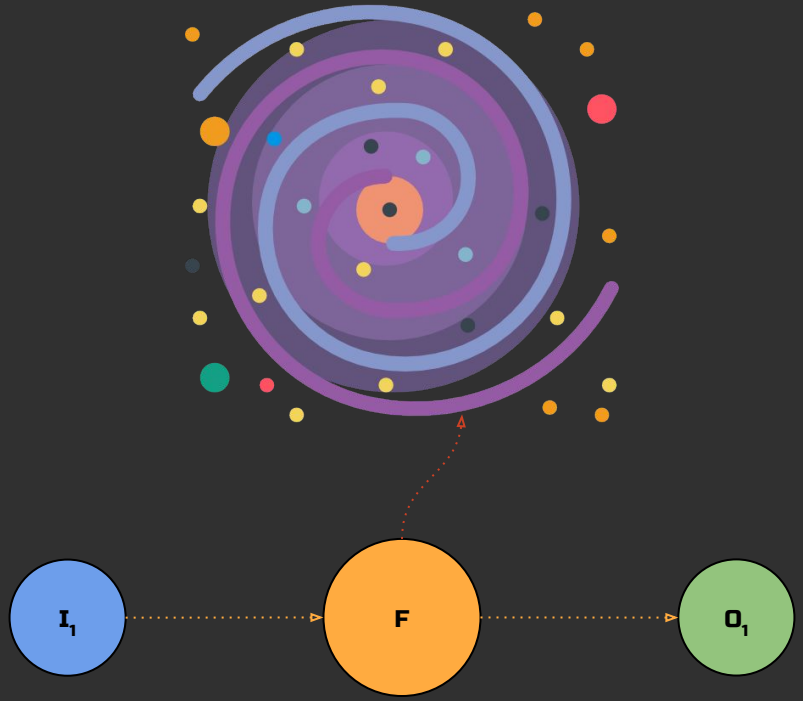


Purity

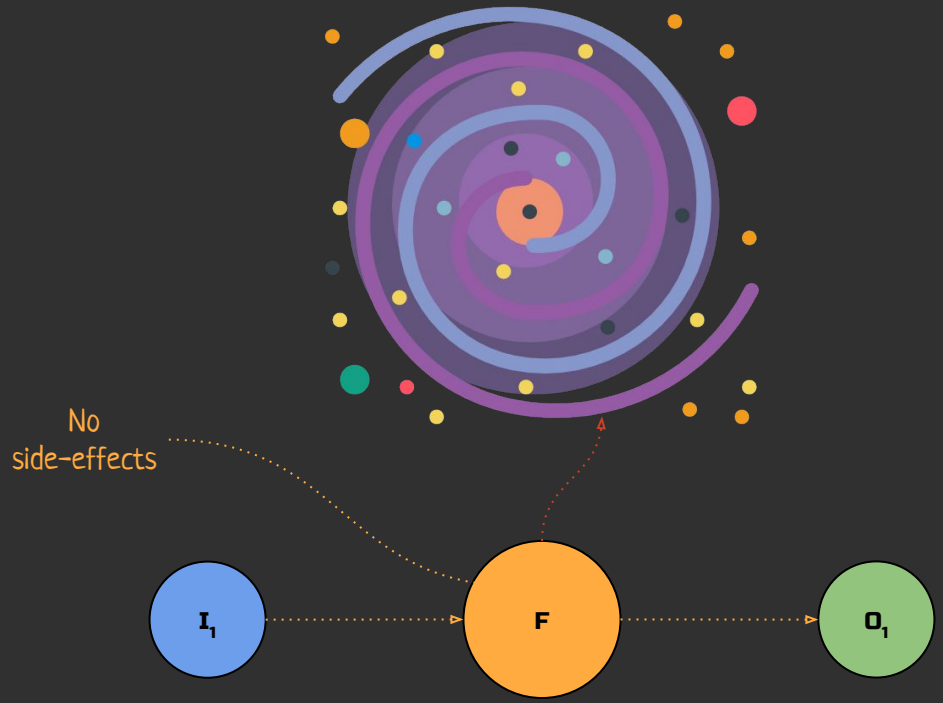




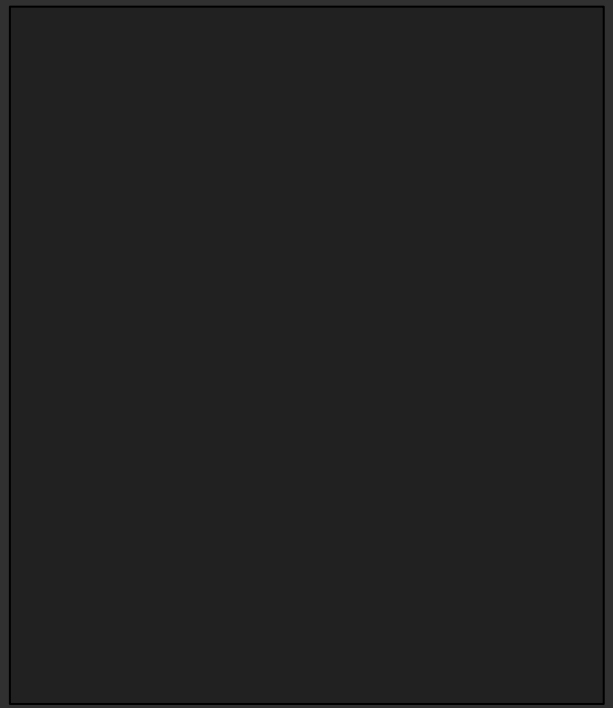
Purity



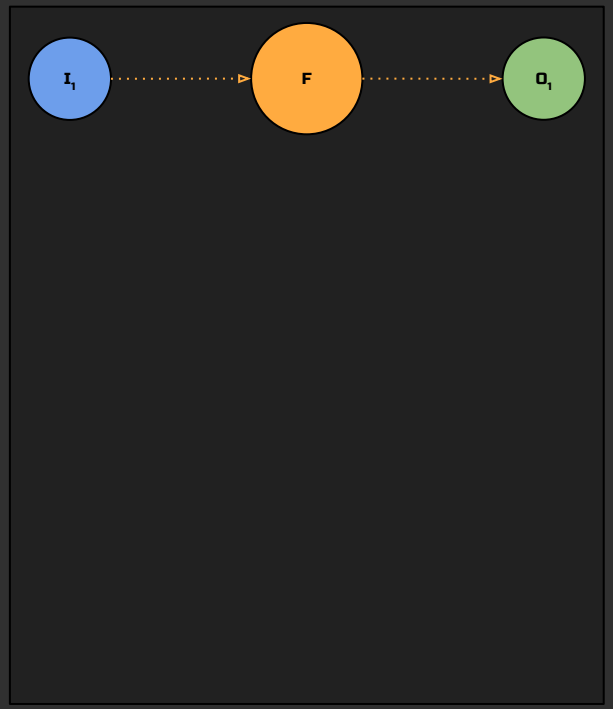
Purity



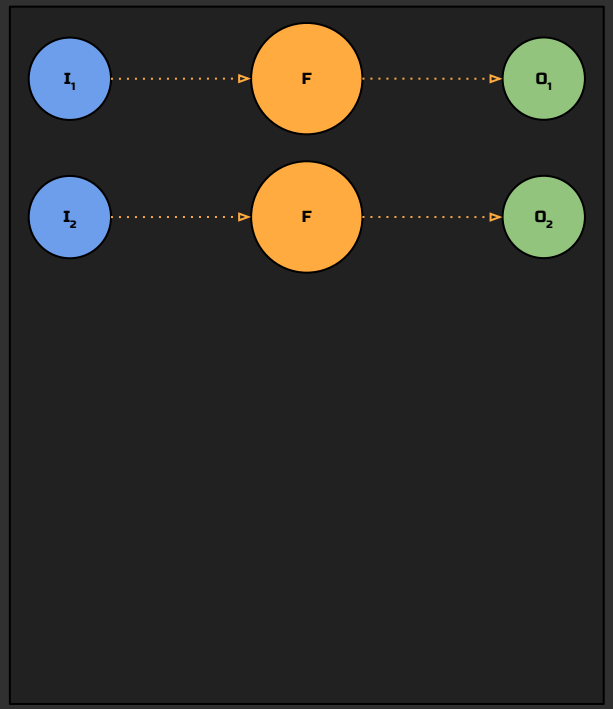
Purity



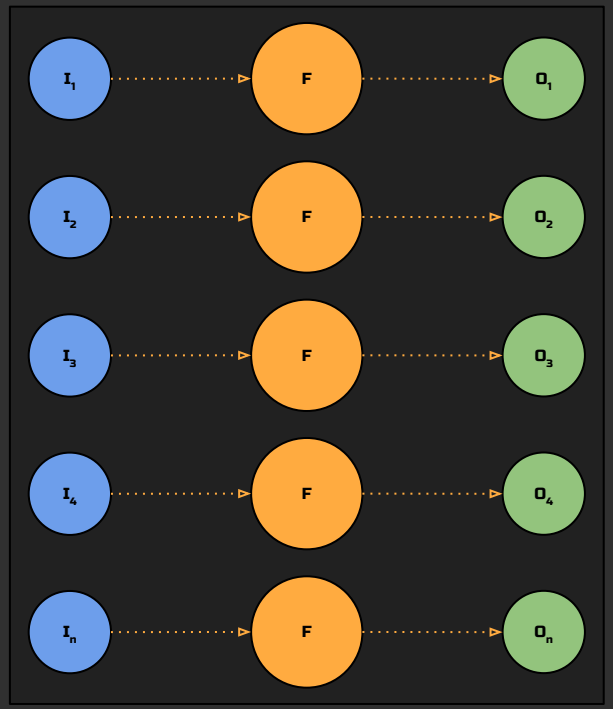
Purity



Purity



Purity



Purity

```
public static int Sum(int a, int b) => a + b;
```

Purity

```
public static int Sum(int a, int b) => a + b;
```

```
public int Method1() => Sum(2, 3);
```

Purity

```
public static int Sum(int a, int b) => a + b;
```

```
public int Method1() => Sum(2, 3);
```

```
public int Method3() => Sum(2, 12);
```

Purity

```
public static int Sum(int a, int b) => a + b;
```

```
public int Method1() => 5;
```

```
public int Method3() => Sum(2, 12);
```

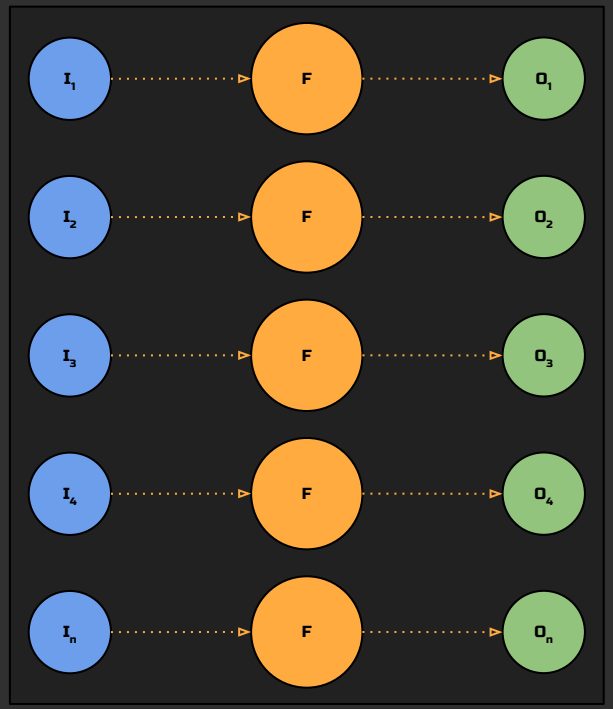
Purity

```
public static int Sum(int a, int b) => a + b;
```

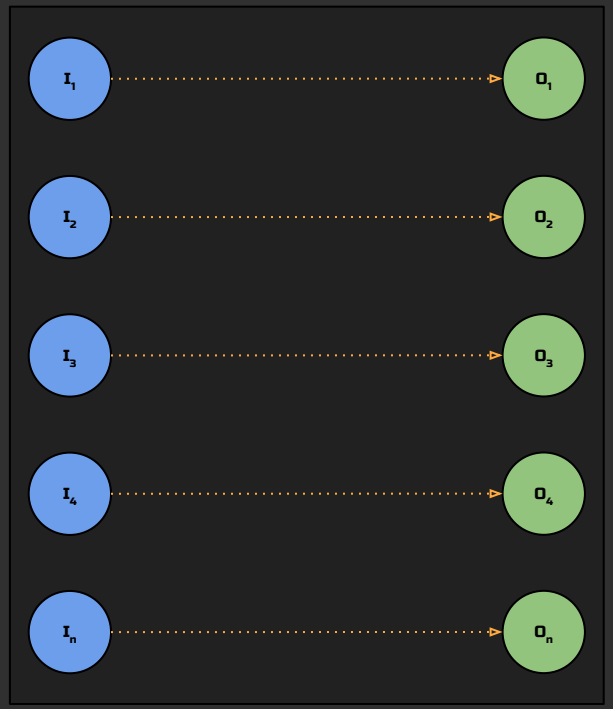
```
public int Method1() => 5;
```

```
public int Method3() => 14;
```

Purity



Purity

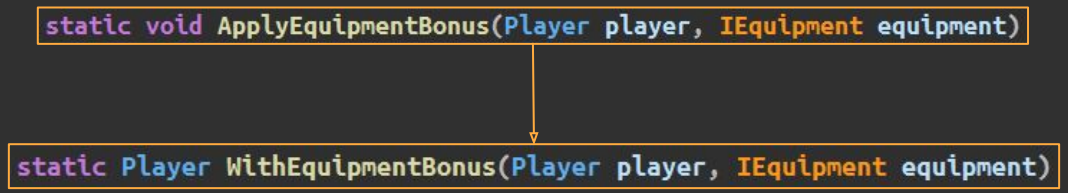


Purity

Immutability

```
static void ApplyEquipmentBonus(Player player, IEquipment equipment)
```

Immutability



Immutability

Mutable

Immutability

Mutable

Жорик

Immutability

Mutable

Жорик

Immutability

Mutable

Жорик

Immutability

Mutable

Жорик

Immutability

Immutability

Immutable

Immutability

Immutable

Жорик1

Immutability

Immutable

Жорик1

Жорик2

Immutability

Immutable

Жорик1

Жорик2

Жорик3

Immutability

Immutable

Жорик1

Жорик2

Жорик3

Жорик4

Immutability

Immutable

Жорик1

Жорик2

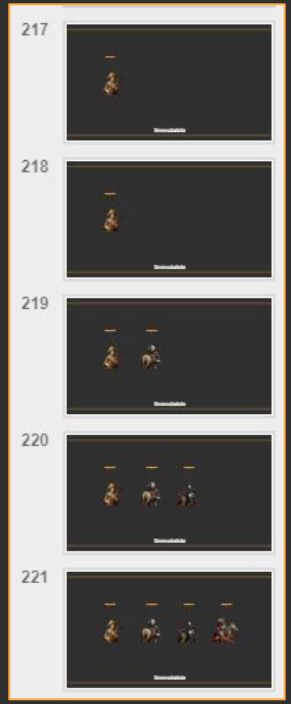
Жорик3

Жорик4

КАК ГИТ!

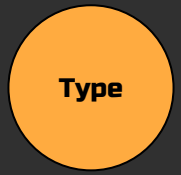
Immutability

Immutable

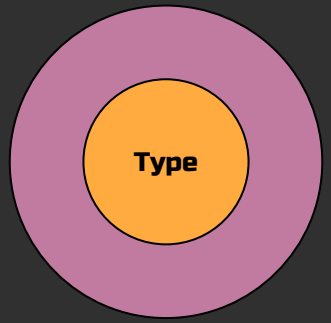


Immutability

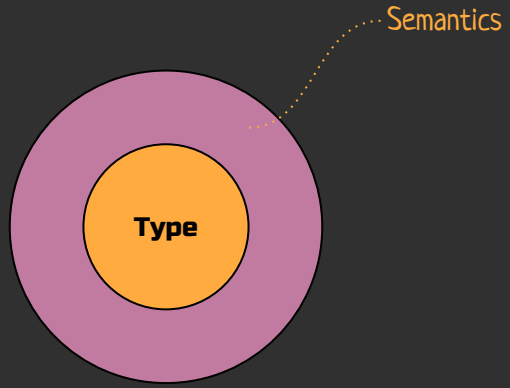
Cool types



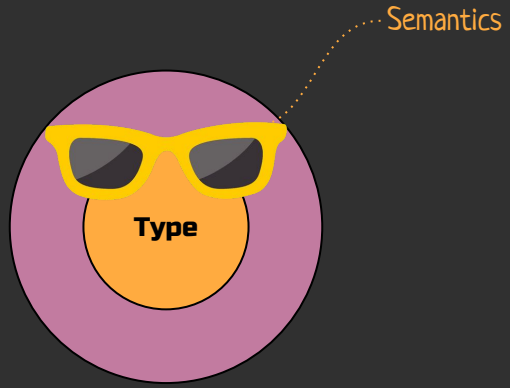
Cool types



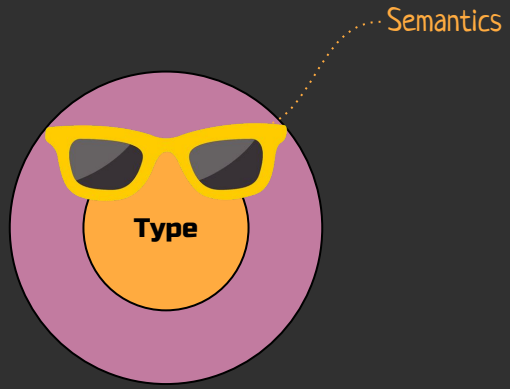
Cool types



Cool types



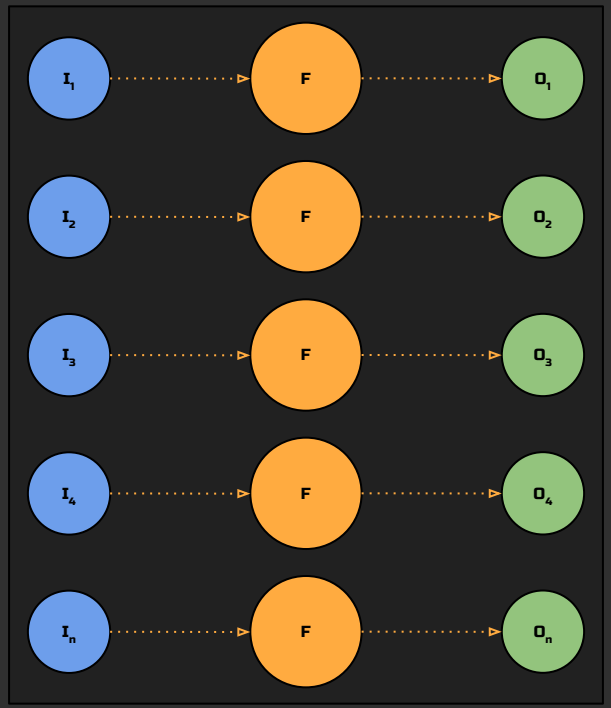
Cool types



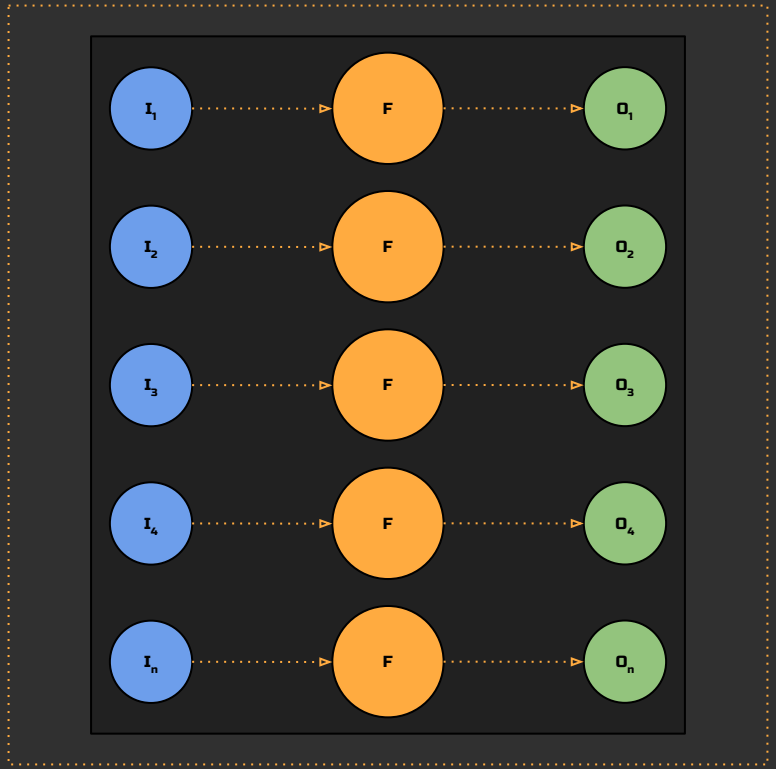

```
public struct Nullable<T> where T : struct
```

Cool types

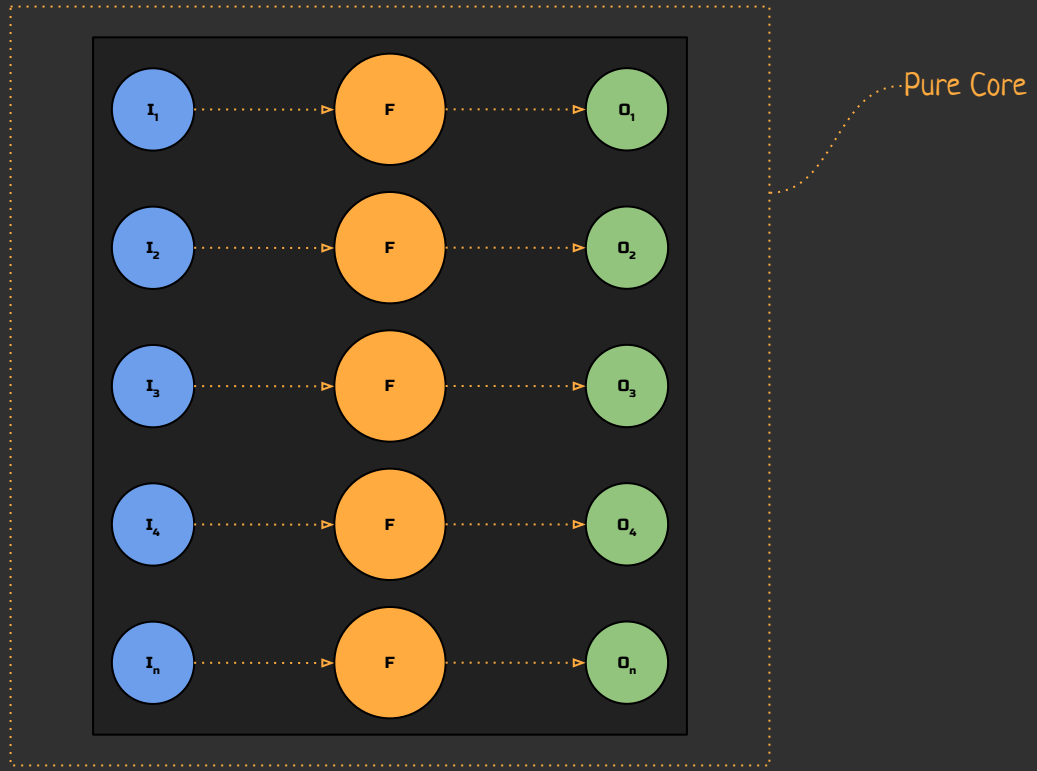
Monads



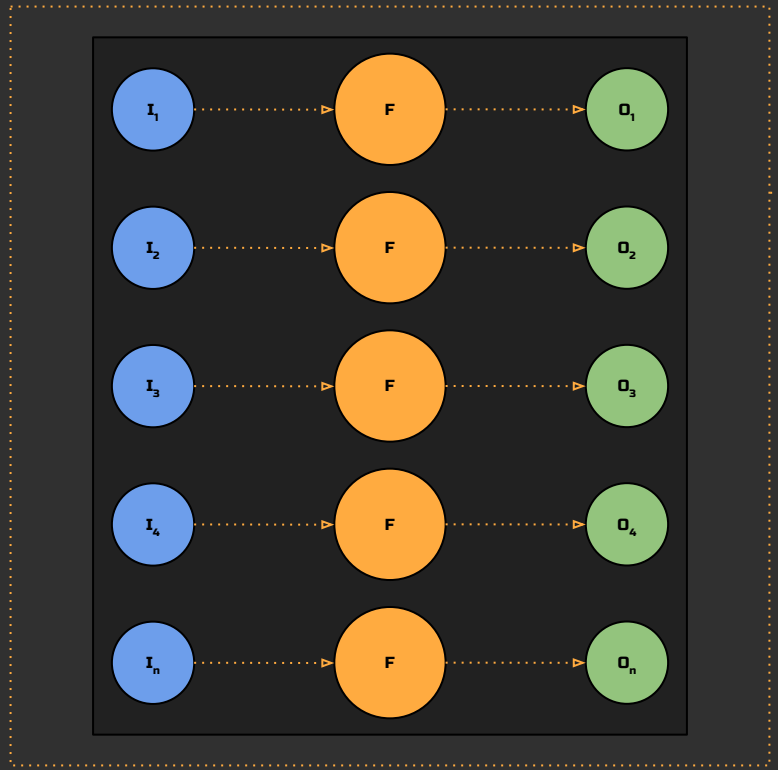
Monads



Monads



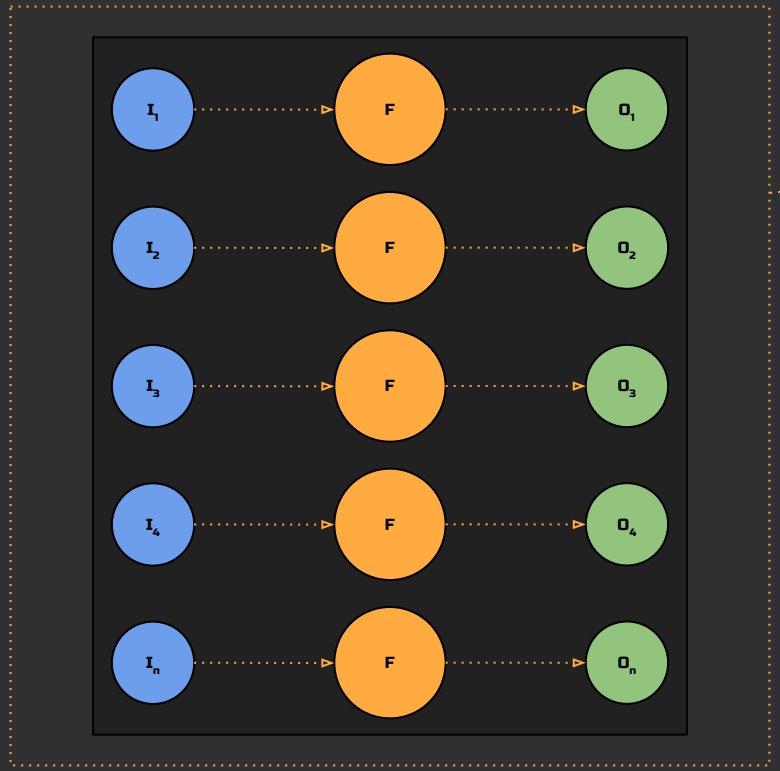
Monads



Monads

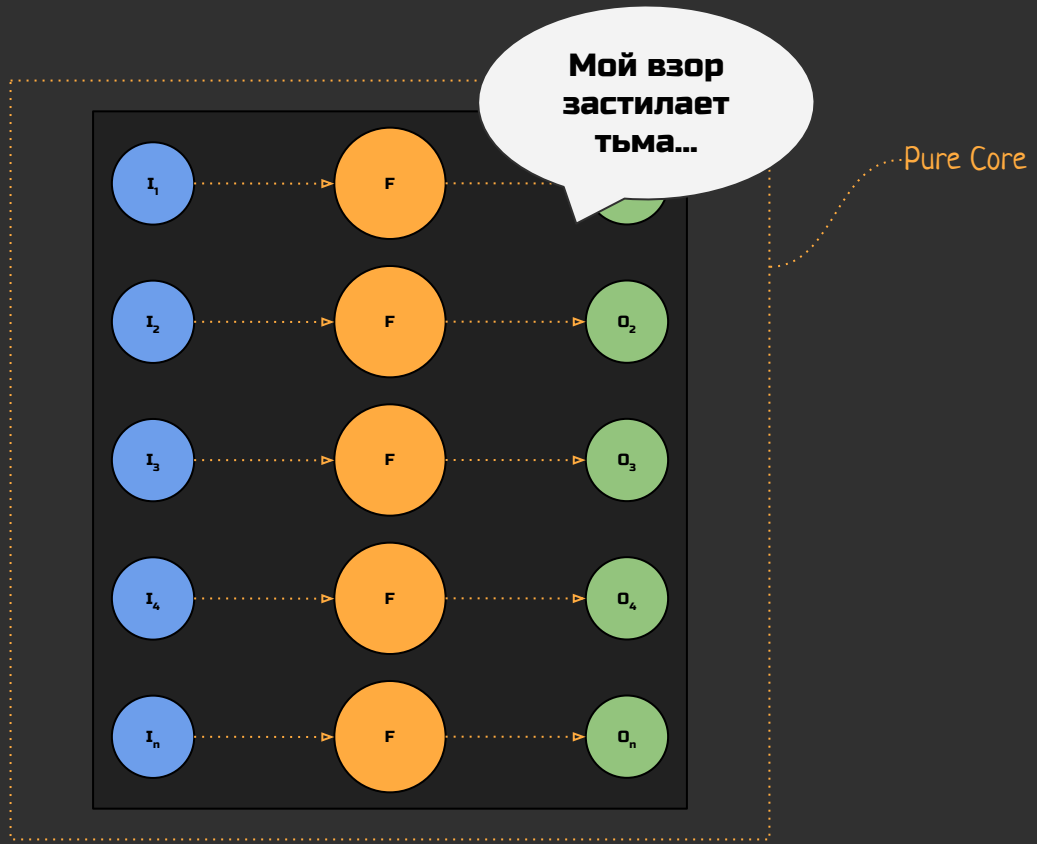
IO

Pure Core



Monads

IO


Мой взор застилает тьма...
Pure Core
I1
I2
I3
I4
In
F
F
F
F
F
O2
O3
O4
On

Monads

Monads

```
public static int ReadNumber()
{



}
```

Monads

```
public static int ReadNumber()
{
  var number = ReadLine();



}
```

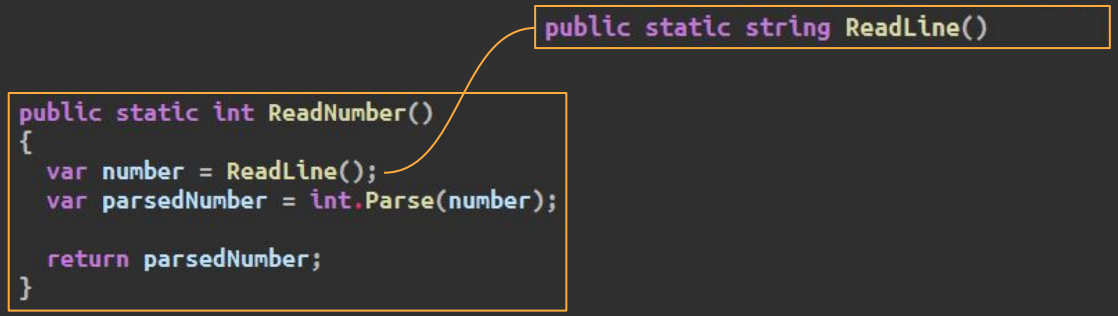
Monads

```
public static int ReadNumber()
{
  var number = ReadLine();
  var parsedNumber = int.Parse(number);


}
```

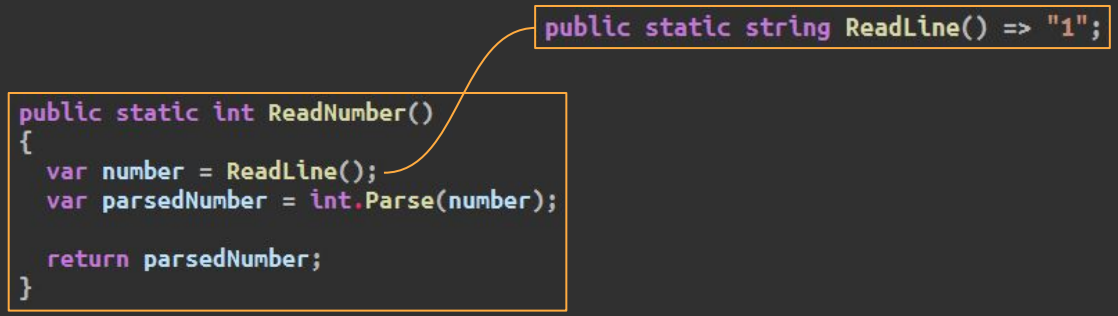
Monads

```
public static int ReadNumber()
{
  var number = ReadLine();
  var parsedNumber = int.Parse(number);

  return parsedNumber;
}
```

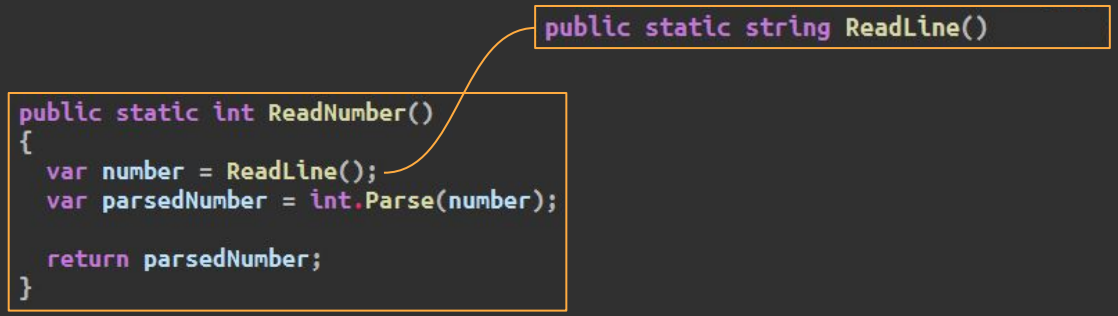
Monads


public static string ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var parsedNumber = int.Parse(number);
return parsedNumber;
}

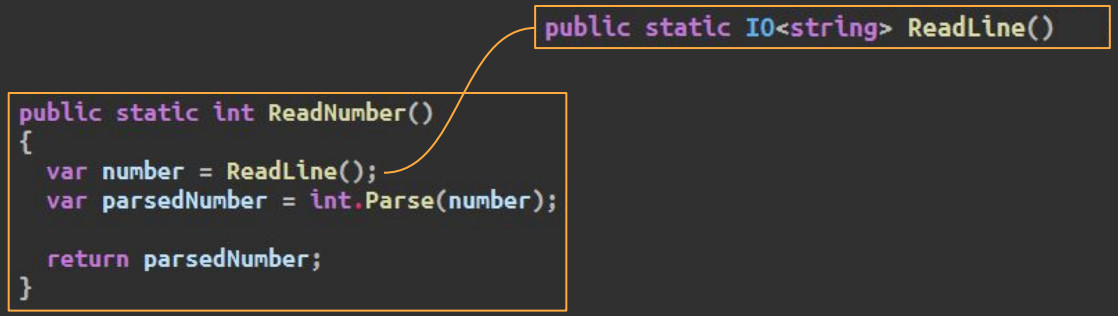
Monads


public static string ReadLine() => "1";
public static int ReadNumber()
{
var number = ReadLine();
var parsedNumber = int.Parse(number);
return parsedNumber;
}

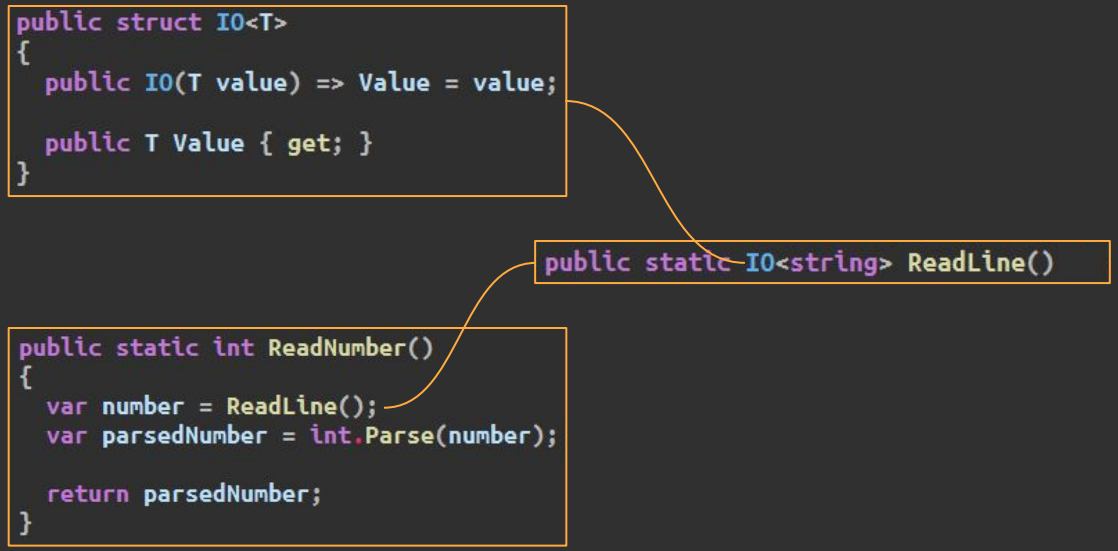
Monads


public static string ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var parsedNumber = int.Parse(number);
return parsedNumber;
}

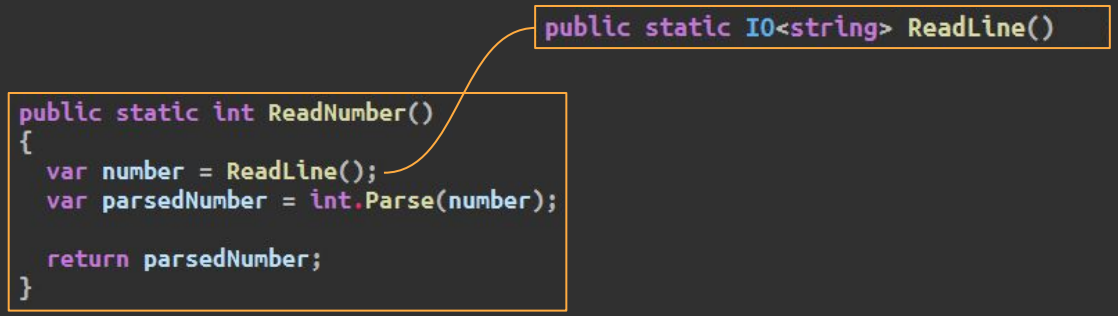
Monads


public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var parsedNumber = int.Parse(number);
return parsedNumber;
}

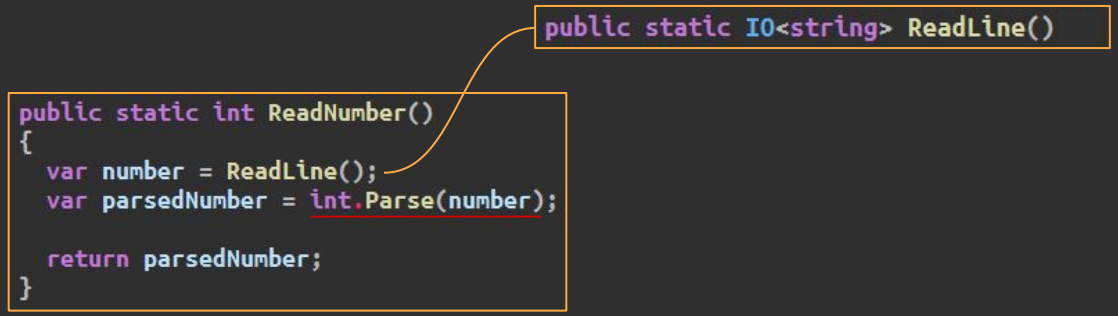
Monads


public struct IO<T>
{
public IO(T value) => Value = value;
public T Value { get; }
}
public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var parsedNumber = int.Parse(number);
return parsedNumber;
}

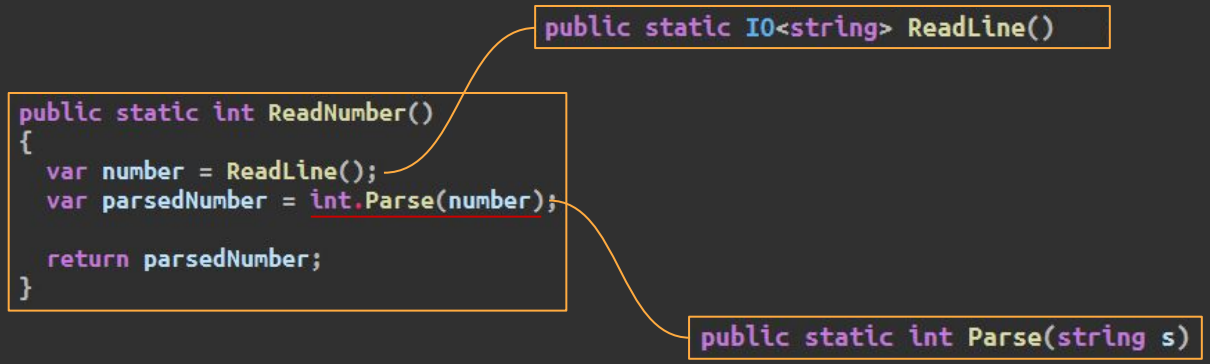
Monads


public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var parsedNumber = int.Parse(number);
return parsedNumber;
}

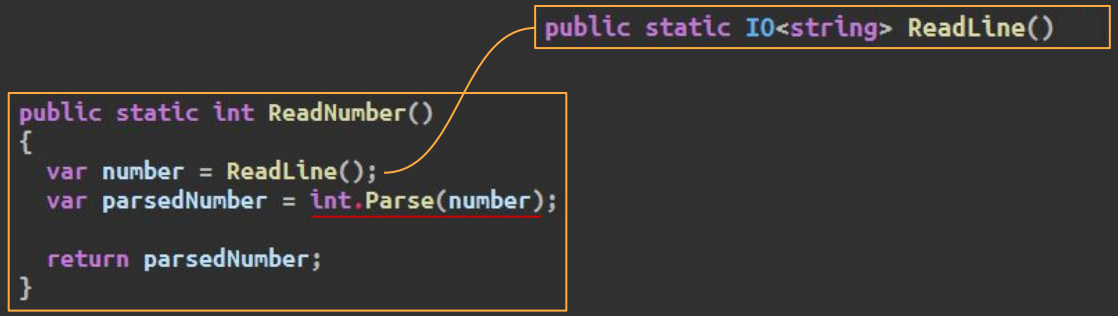
Monads


public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var parsedNumber = int.Parse(number);
return parsedNumber;
}

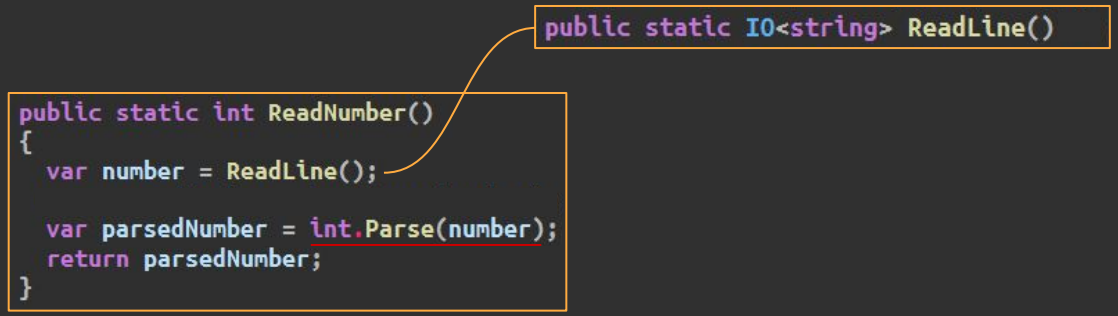
Monads


public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var parsedNumber = int.Parse(number);
return parsedNumber;
}
public static int Parse(string s)

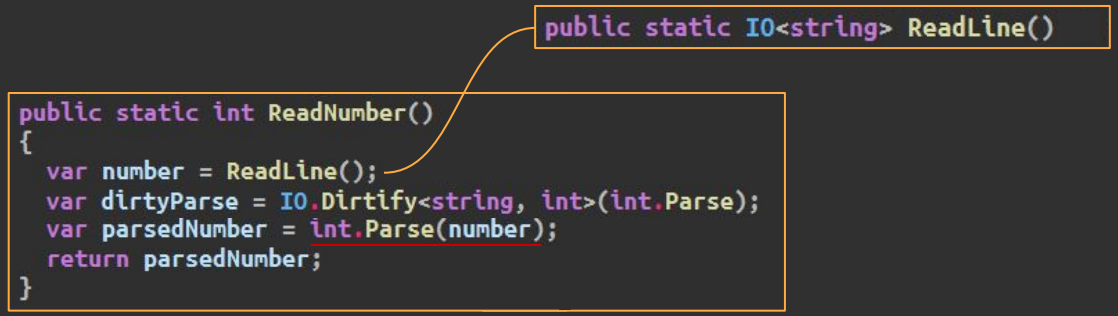
Monads


public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var parsedNumber = int.Parse(number);
return parsedNumber;
}

Monads


public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var parsedNumber = int.Parse(number);
return parsedNumber;
}

Monads


public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var dirtyParse = IO.Dirtify<string, int>(int.Parse);
var parsedNumber = int.Parse(number);
return parsedNumber;
}

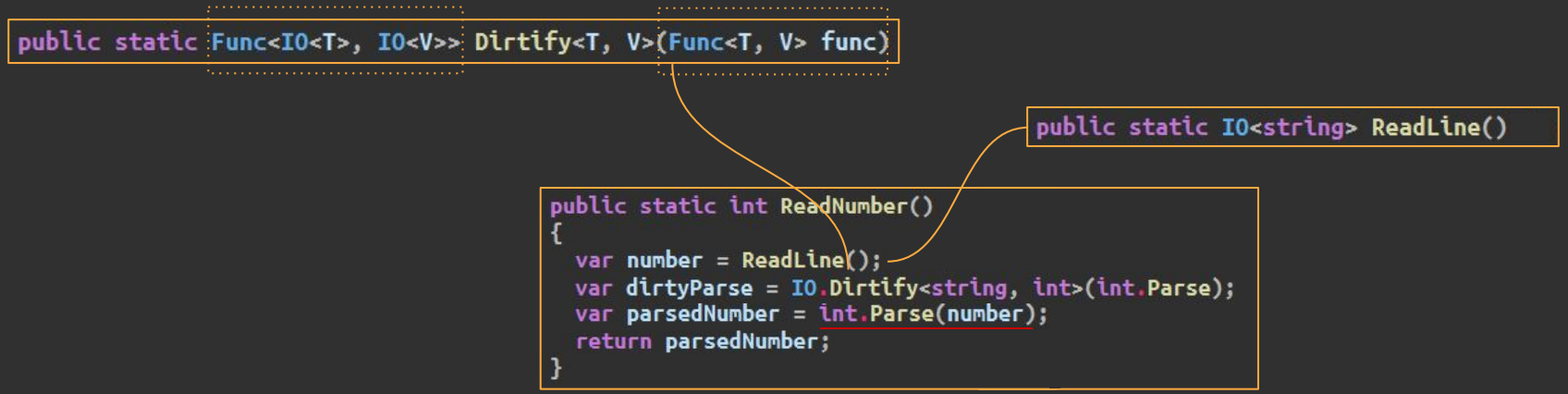
Monads

```
public static Func<IO<T>, IO<V>> Dirtify<T, V>(Func<T, V> func)
```

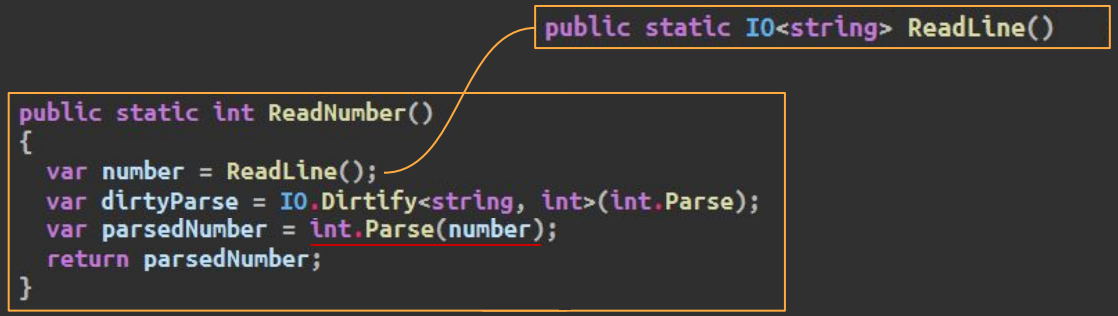
```
public static IO<string> ReadLine()
```

```
public static int ReadNumber()
{
  var number = ReadLine();
  var dirtyParse = IO.Dirtify<string, int>(int.Parse);
  var parsedNumber = int.Parse(number);
  return parsedNumber;
}
```

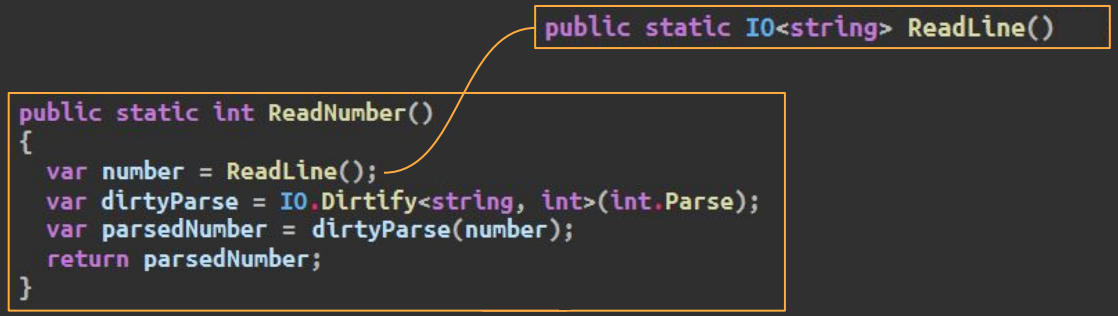
Monads


public static Func<IO<T>, IO<V>> Dirtify<T, V>(Func<T, V> func)
public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var dirtyParse = IO.Dirtify<string, int>(int.Parse);
var parsedNumber = int.Parse(number);
return parsedNumber;
}

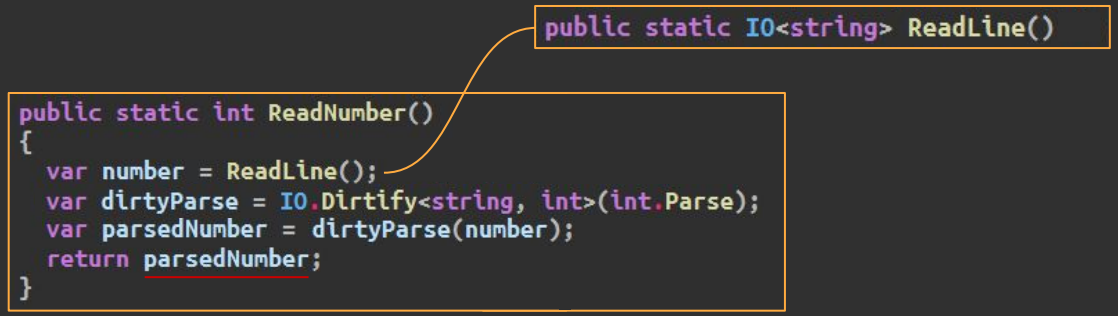
Monads


public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var dirtyParse = IO.Dirtify<string, int>(int.Parse);
var parsedNumber = int.Parse(number);
return parsedNumber;
}

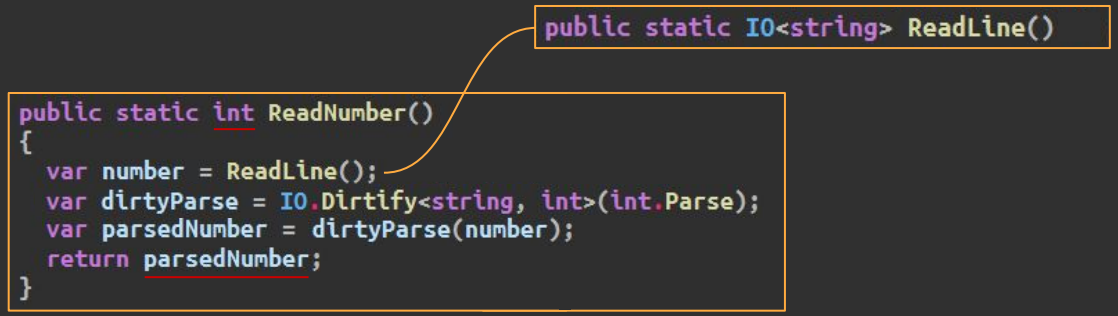
Monads


public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var dirtyParse = IO.Dirtify<string, int>(int.Parse);
var parsedNumber = dirtyParse(number);
return parsedNumber;
}

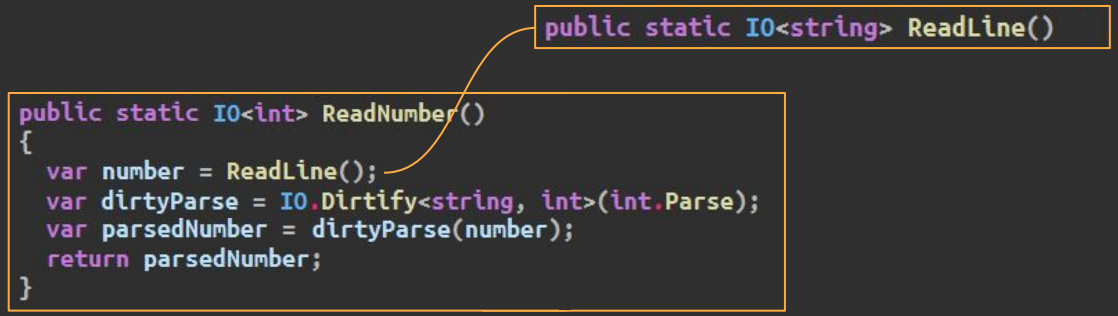
Monads


public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var dirtyParse = IO.Dirtify<string, int>(int.Parse);
var parsedNumber = dirtyParse(number);
return parsedNumber;
}

Monads

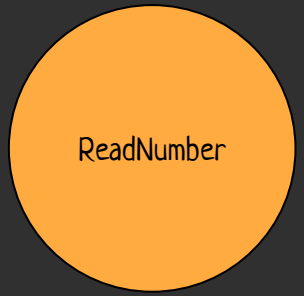

public static IO<string> ReadLine()
public static int ReadNumber()
{
var number = ReadLine();
var dirtyParse = IO.Dirtify<string, int>(int.Parse);
var parsedNumber = dirtyParse(number);
return parsedNumber;
}

Monads

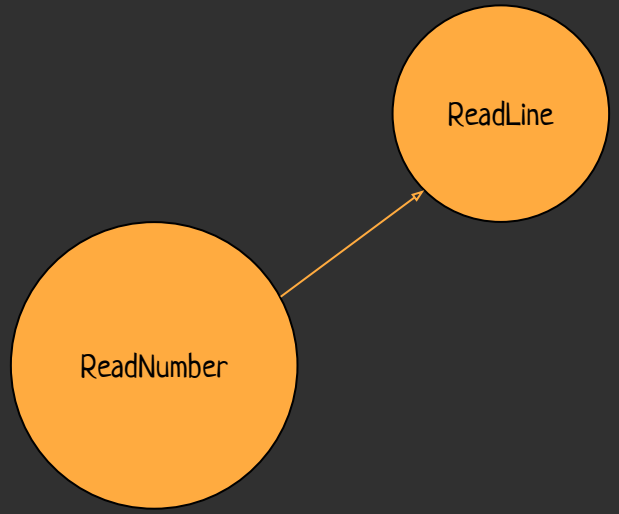

public static IO<string> ReadLine()
public static IO<int> ReadNumber()
{
var number = ReadLine();
var dirtyParse = IO.Dirtify<string, int>(int.Parse);
var parsedNumber = dirtyParse(number);
return parsedNumber;
}

Monads

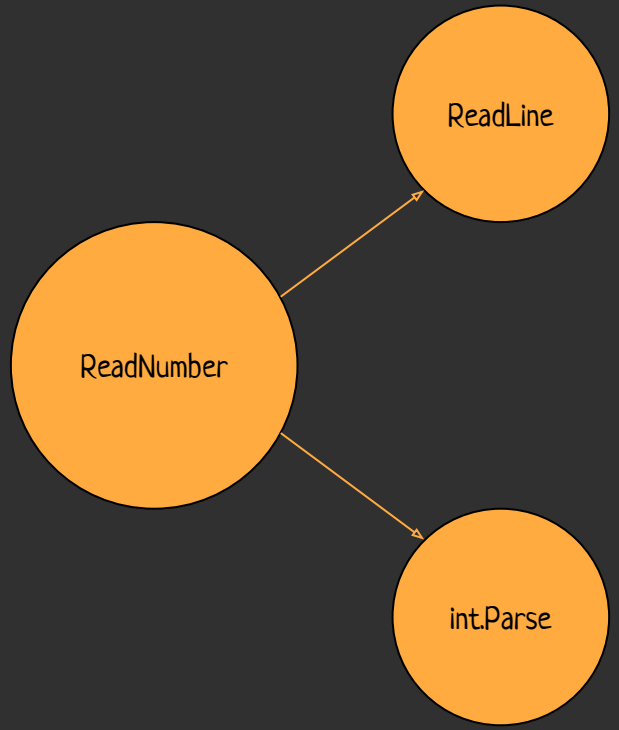
Monads



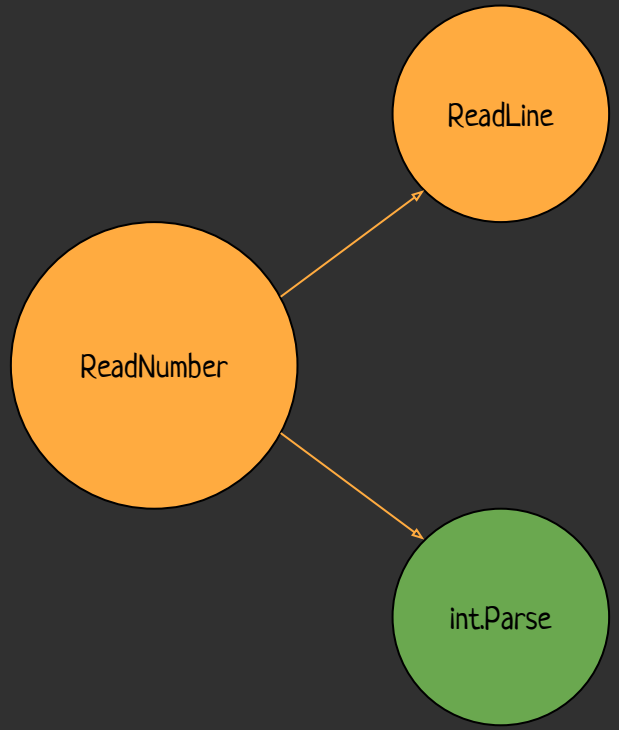
Monads



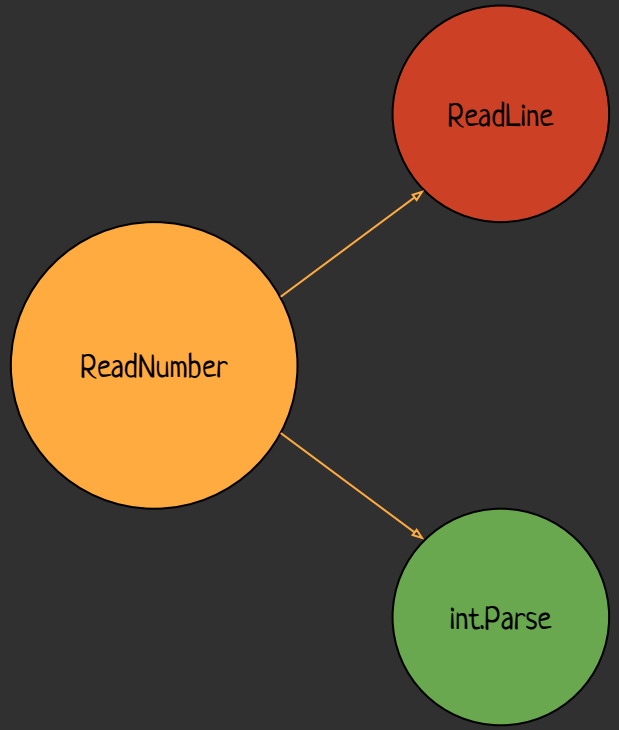
Monads



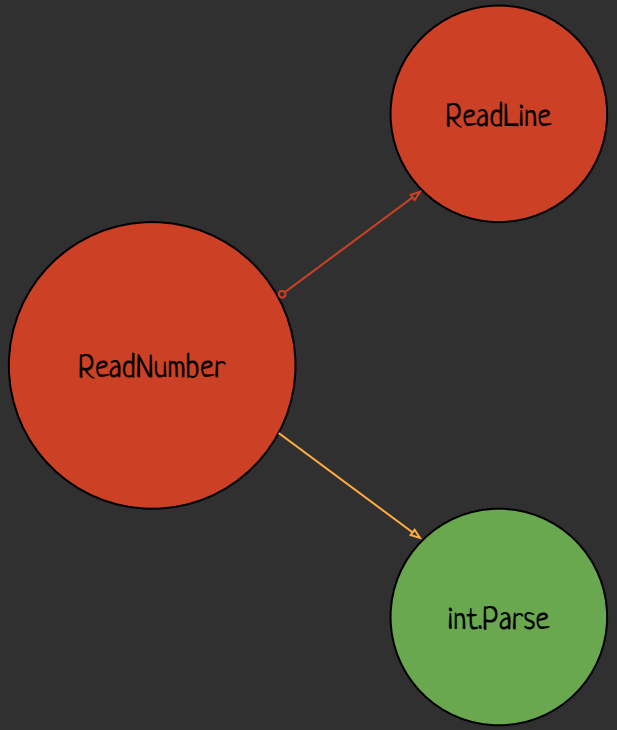
Monads



Monads



Monads

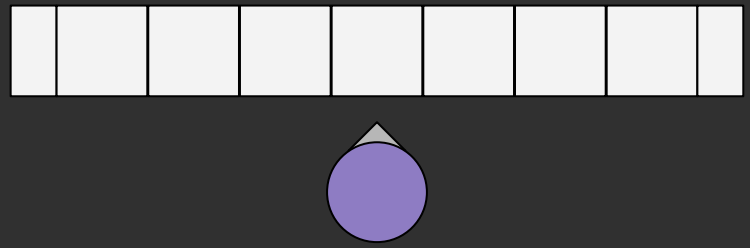


Monads

Declarative vs Imperative

```
F, A, B ::= x
          | (λx.B)
          | (F A)
```

vs

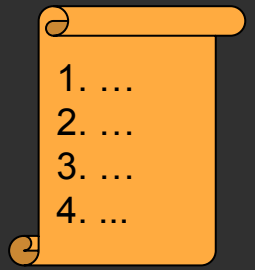


q_1	x_0	a_0	m_0	q_m
q_1	x_1	a_1	m_1	q_n
...	...	...	...	...

Declarative vs Imperative

```
f => f(f)
```

vs



Declarative vs Imperative

What vs How

Declarative vs Imperative

C#

LINQ

```
var allSlots:Array = InventoryManager.getAllSlots();
var userHasNoItems:Boolean = query(allSlots)
  .where(function (u:UserInventorySlot):Boolean { return !u.locked && u.item != null })
  .where(function (u:UserInventorySlot):Boolean { return !u.item.inventoryItemInfo.sealed })
  .toArray().length == 0;
```

LINQ

```
private Dictionary<SceneObjectTypeId, int> Objects()
{
  return _instance.Balance
    .Select(x => new { TypeId = x.Key, Count = x.Value })
    .Concat(
      _instance
        .Positions()
        .Where(x => x.IsStatic)
        .SelectMany(x => x.StaticInfo.Objects)
        .Select(x => x.TypeId)
        .GroupBy(x => x)
        .Select(x => new { TypeId = x.Key, Count = x.Count() }))
    .GroupBy(x => x.TypeId)
    .ToDictionary(x => x.Key, x => x.Sum(y => y.Count));
}
```

LINQ

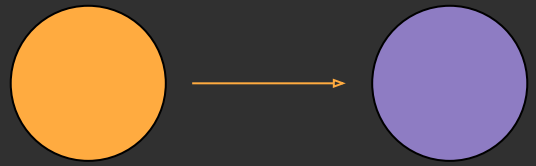
Expression that is composition of functions

```
private Dictionary<SceneObjectTypeId, int> Objects()
{
  return _instance.Balance
    .Select(x => new { TypeId = x.Key, Count = x.Value })
    .Concat(
      _instance
        .Positions()
        .Where(x => x.IsStatic)
        .SelectMany(x => x.StaticInfo.Objects)
        .Select(x => x.TypeId)
        .GroupBy(x => x)
        .Select(x => new { TypeId = x.Key, Count = x.Count() }))
    .GroupBy(x => x.TypeId)
    .ToDictionary(x => x.Key, x => x.Sum(y => y.Count));
}
```

LINQ

As (Map)

As (Map)



```
public static TOut As<TIn, TOut>(this TIn self, Func<TIn, TOut> map) =>
  map(self);
```

As (Map)

```
public static string Md5(this StringBuilder self)
{



}
```

As (Map)

```
public static string Md5(this StringBuilder self)
{
  var str = self.ToString();




}
```

As (Map)

```
public static string Md5(this StringBuilder self)
{
  var str = self.ToString();
  var bytes = Encoding.UTF8.GetBytes(str);



}
```

As (Map)

```
public static string Md5(this StringBuilder self)
{
  var str = self.ToString();
  var bytes = Encoding.UTF8.GetBytes(str);

  using (var crypto = new MD5CryptoServiceProvider())
  {




  }
}
```

As (Map)

```
public static string Md5(this StringBuilder self)
{
  var str = self.ToString();
  var bytes = Encoding.UTF8.GetBytes(str);

  using (var crypto = new MD5CryptoServiceProvider())
  {
    var hash = crypto.ComputeHash(bytes);



  }
}
```

As (Map)

```
public static string Md5(this StringBuilder self)
{
  var str = self.ToString();
  var bytes = Encoding.UTF8.GetBytes(str);

  using (var crypto = new MD5CryptoServiceProvider())
  {
    var hash = crypto.ComputeHash(bytes);
    var builder = new StringBuilder(hash.Length * 2);



  }
}
```

As (Map)

```
public static string Md5(this StringBuilder self)
{
  var str = self.ToString();
  var bytes = Encoding.UTF8.GetBytes(str);

  using (var crypto = new MD5CryptoServiceProvider())
  {
    var hash = crypto.ComputeHash(bytes);
    var builder = new StringBuilder(hash.Length * 2);

    builder.AppendFormattedSequence(hash, "{0:x2}");

  }
}
```

As (Map)

```
public static string Md5(this StringBuilder self)
{
  var str = self.ToString();
  var bytes = Encoding.UTF8.GetBytes(str);

  using (var crypto = new MD5CryptoServiceProvider())
  {
    var hash = crypto.ComputeHash(bytes);
    var builder = new StringBuilder(hash.Length * 2);

    builder.AppendFormattedSequence(hash, "{0:x2}");
    return builder.ToString();
  }
}
```

As (Map)

```
public static string Md5(this StringBuilder self) => self
```

As (Map)

```
public static string Md5(this StringBuilder self) => self
  .As(x => x.ToString())
```

As (Map)

```
public static string Md5(this StringBuilder self) => self
  .As(x => x.ToString())
  .As(Encoding.UTF8.GetBytes)
```

As (Map)

```
public static string Md5(this StringBuilder self) => self
  .As(x => x.ToString())
  .As(Encoding.UTF8.GetBytes)
  .As(x => new MD5CryptoServiceProvider().Using(y => y.ComputeHash(x)))
```

As (Map)

```
public static TOut Using<TIn, TOut>(this TIn self, Func<TIn, TOut> map) where TIn : IDisposable
{
  var result = map(self);
  self.Dispose();
  return result;
}
```

```
public static string Md5(this StringBuilder self) => self
  .As(x => x.ToString())
  .As(Encoding.UTF8.GetBytes)
  .As(x => new MD5CryptoServiceProvider().Using(y => y.ComputeHash(x)))
```

As (Map)

```
public static string Md5(this StringBuilder self) => self
  .As(x => x.ToString())
  .As(Encoding.UTF8.GetBytes)
  .As(x => new MD5CryptoServiceProvider().Using(y => y.ComputeHash(x)))
```

As (Map)

```
public static string Md5(this StringBuilder self) => self
  .As(x => x.ToString())
  .As(Encoding.UTF8.GetBytes)
  .As(x => new MD5CryptoServiceProvider().Using(y => y.ComputeHash(x)))
  .As(x => new StringBuilder(x.Length * 2)
```

As (Map)

```
public static string Md5(this StringBuilder self) => self
  .As(x => x.ToString())
  .As(Encoding.UTF8.GetBytes)
  .As(x => new MD5CryptoServiceProvider().Using(y => y.ComputeHash(x)))
  .As(x => new StringBuilder(x.Length * 2)
    .AppendFormattedSequence(x, "{0:x2}"))
```

As (Map)

```
public static string Md5(this StringBuilder self) => self
  .As(x => x.ToString())
  .As(Encoding.UTF8.GetBytes)
  .As(x => new MD5CryptoServiceProvider().Using(y => y.ComputeHash(x)))
  .As(x => new StringBuilder(x.Length * 2)
    .AppendFormattedSequence(x, "{0:x2}"))
  .ToString();
```

As (Map)

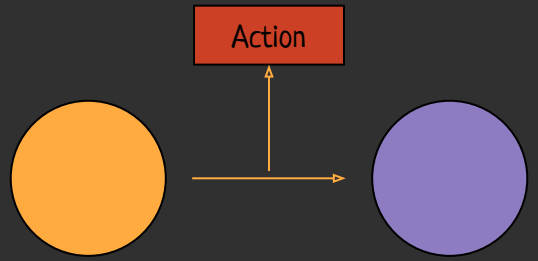
```
public static string Md5(this StringBuilder self) => self
  .As(x => x.ToString())
  .As(Encoding.UTF8.GetBytes)
  .As(x => new MD5CryptoServiceProvider().Using(y => y.ComputeHash(x)))
  .As(x => new StringBuilder(x.Length * 2)
    .AppendFormattedSequence(x, "{0:x2}"))
  .ToString();
```

```
public static string Md5(this StringBuilder self)
{
  var str = self.ToString();
  var bytes = Encoding.UTF8.GetBytes(str);

  using (var crypto = new MD5CryptoServiceProvider())
  {
    var hash = crypto.ComputeHash(bytes);
    var builder = new StringBuilder(hash.Length * 2);

    builder.AppendFormattedSequence(hash, "{0:x2}");
    return builder.ToString();
  }
}
```

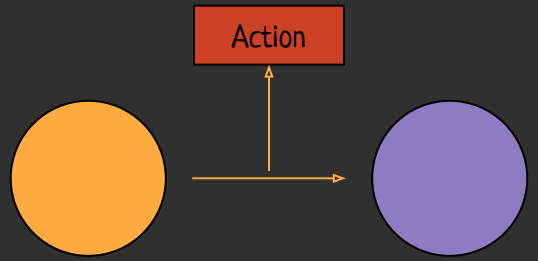
As (Map)

Do (Tee)

Do (Tee)

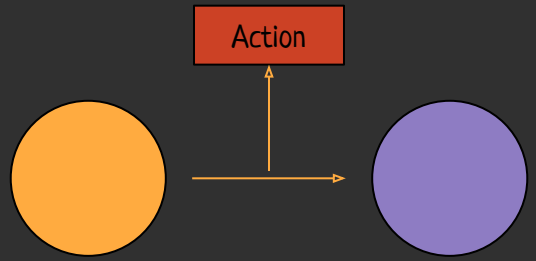


Do (Tee)




```
public static T Do<T>(this T self, Action<T> action)
{
  if (self != null)
    action(self);

  return self;
}
```

Do (Tee)



Side-effect

```
public static T Do<T>(this T self, Action<T> action)
{
  if (self != null)
    action(self);

  return self;
}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{



}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  var jsonStr = jsonObject.ToJsonStr();



}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  var jsonStr = jsonObject.ToJsonStr();
  var jsonBytes = Encoding.UTF8.GetBytes(jsonStr);




}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  var jsonStr = jsonObject.ToJsonStr();
  var jsonBytes = Encoding.UTF8.GetBytes(jsonStr);

  self.ContentLength = jsonBytes.Length;




}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  var jsonStr = jsonObject.ToJsonStr();
  var jsonBytes = Encoding.UTF8.GetBytes(jsonStr);

  self.ContentLength = jsonBytes.Length;

  using (var stream = self.GetRequestStream())



}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  var jsonStr = jsonObject.ToJsonStr();
  var jsonBytes = Encoding.UTF8.GetBytes(jsonStr);

  self.ContentLength = jsonBytes.Length;

  using (var stream = self.GetRequestStream())
    stream.Write(jsonBytes, 0, jsonBytes.Length);


}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  var jsonStr = jsonObject.ToJsonStr();
  var jsonBytes = Encoding.UTF8.GetBytes(jsonStr);

  self.ContentLength = jsonBytes.Length;

  using (var stream = self.GetRequestStream())
    stream.Write(jsonBytes, 0, jsonBytes.Length);

  return self;
}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  jsonObject
}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  jsonObject
    .ToJsonStr()




}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  jsonObject
    .ToJsonStr()
    .As(Encoding.UTF8.GetBytes)



}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  jsonObject
    .ToJsonStr()
    .As(Encoding.UTF8.GetBytes)
    .Do(x => self.ContentLength = x.Length)



}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  jsonObject
    .ToJsonStr()
    .As(Encoding.UTF8.GetBytes)
    .Do(x => self.ContentLength = x.Length)
    .Do(x => self
      .GetRequestStream()



}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  jsonObject
    .ToJsonStr()
    .As(Encoding.UTF8.GetBytes)
    .Do(x => self.ContentLength = x.Length)
    .Do(x => self
      .GetRequestStream()
      .Using(stream => stream.Write(x, 0, x.Length)));


}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  jsonObject
    .ToJsonStr()
    .As(Encoding.UTF8.GetBytes)
    .Do(x => self.ContentLength = x.Length)
    .Do(x => self
      .GetRequestStream()
      .Using(stream => stream.Write(x, 0, x.Length)));

  return self;
}
```

Do (Tee)

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  jsonObject
    .ToJsonStr()
    .As(Encoding.UTF8.GetBytes)
    .Do(x => self.ContentLength = x.Length)
    .Do(x => self
      .GetRequestStream()
      .Using(stream => stream.Write(x, 0, x.Length)));

  return self;
}
```

```
public static HttpWebRequest WithJson<T>(this HttpWebRequest self, T jsonObject)
{
  var jsonStr = jsonObject.ToJsonStr();
  var jsonBytes = Encoding.UTF8.GetBytes(jsonStr);

  self.ContentLength = jsonBytes.Length;

  using (var stream = self.GetRequestStream())
    stream.Write(jsonBytes, 0, jsonBytes.Length);

  return self;
}
```

Do (Tee)

Cool types

```
public static int Divide(int a, int b) => a / b;
```

Cool types

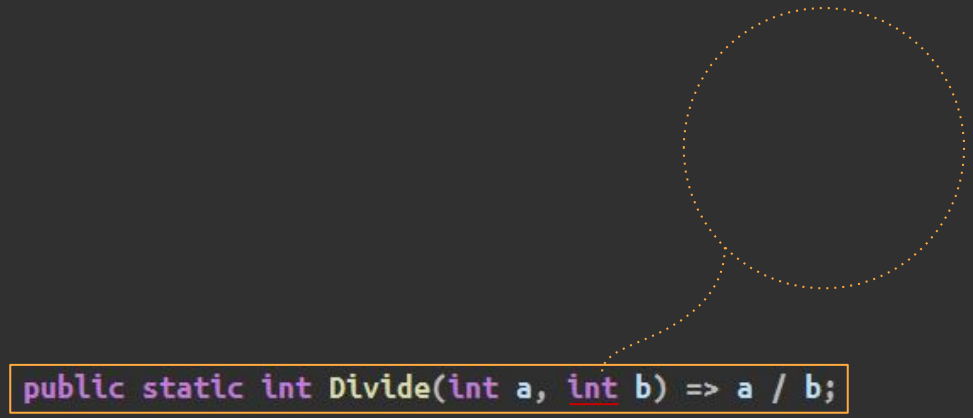
Fair enough?

```
public static int Divide(int a, int b) => a / b;
```

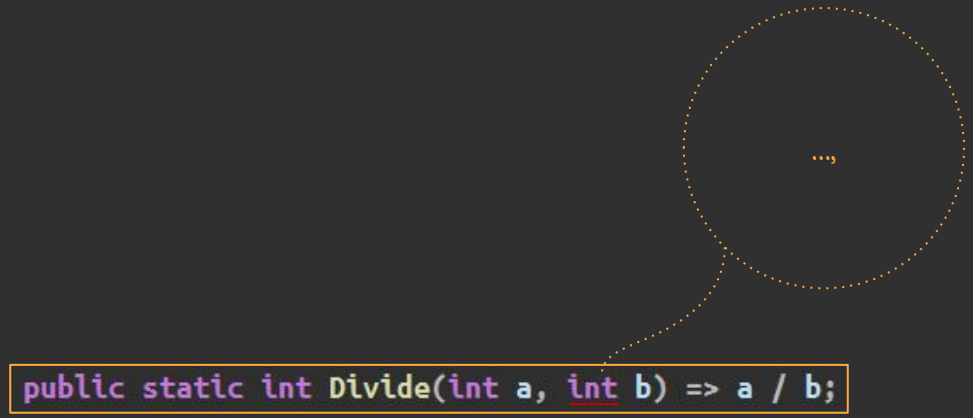
Cool types

```
public static int Divide(int a, int b) => a / b;
```

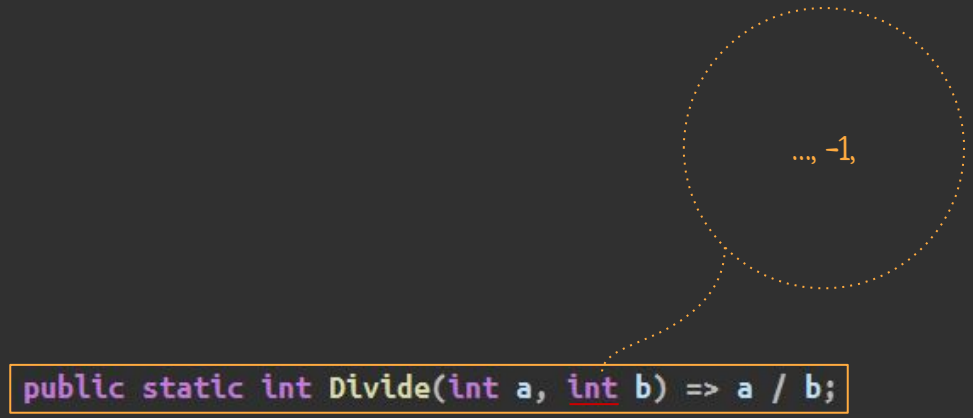
Cool types



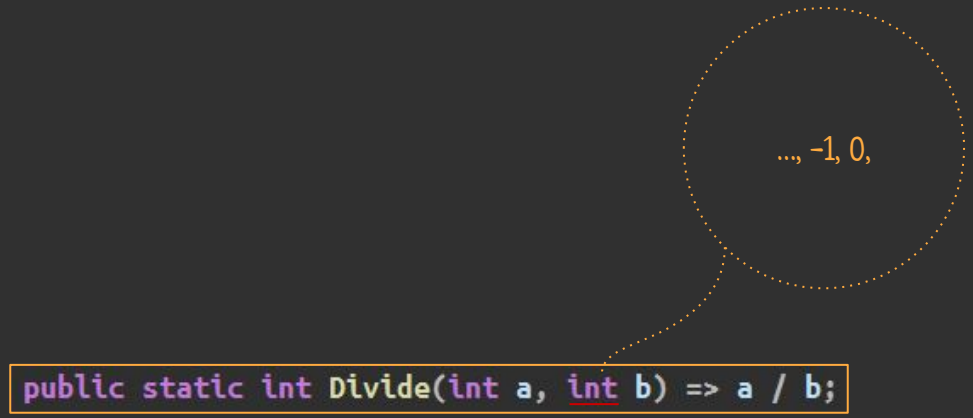
Cool types



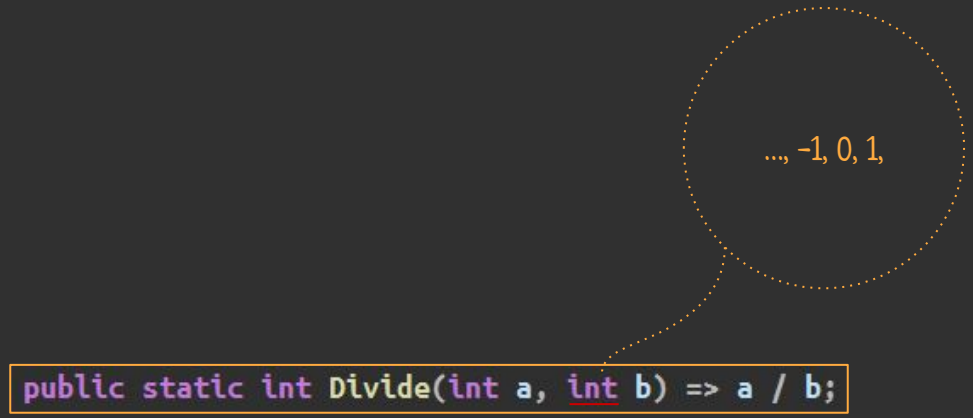
Cool types



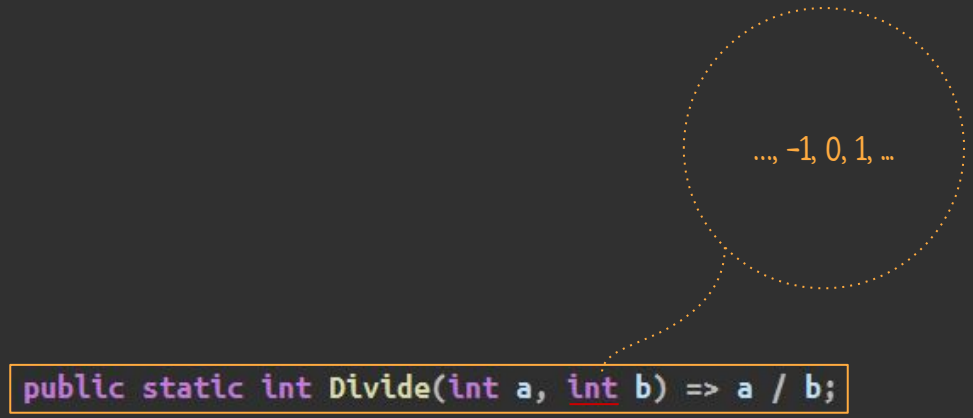
Cool types



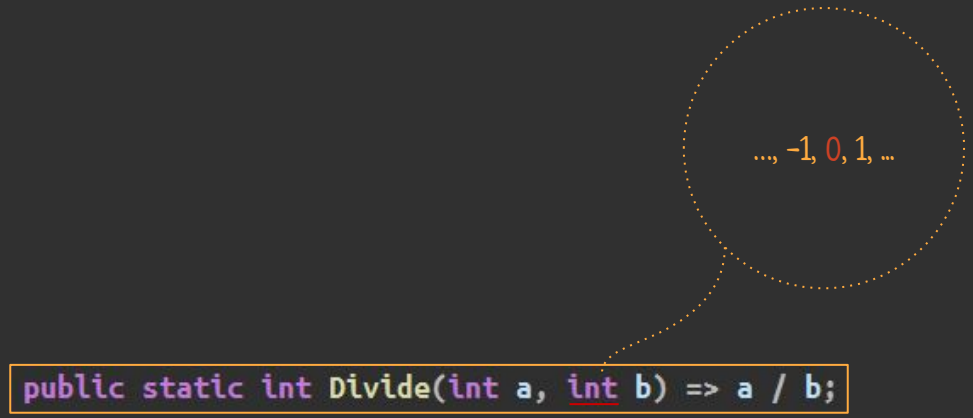
Cool types



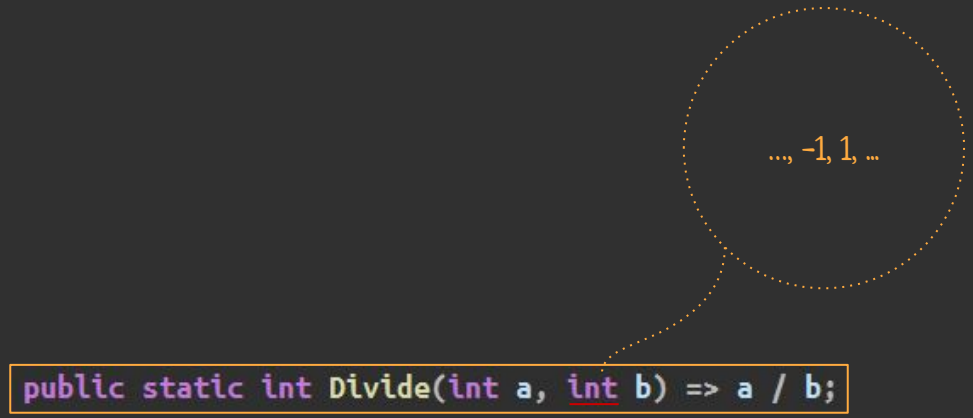
Cool types



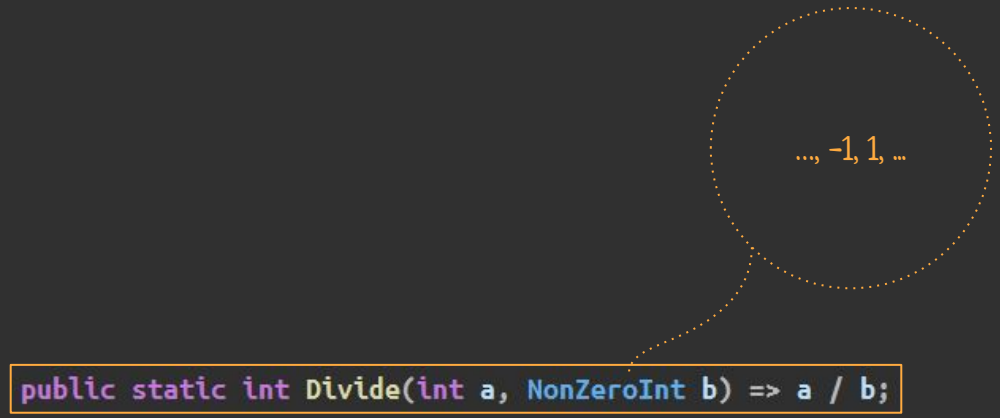
Cool types



Cool types



Cool types



Cool types

```
public struct NonZeroInt
{
  public NonZeroInt(int value)
  {
    if (value == 0)
      throw new Exception();

    Value = value;
  }

  public int Value { get; }

  public static implicit operator int(NonZeroInt self) => self.Value;
}
```


```
public static int Divide(int a, NonZeroInt b) => a / b;
```

Cool types

Cool types

Maybe monad

```
public interface IRepository
{
  User User(long id);
}
```

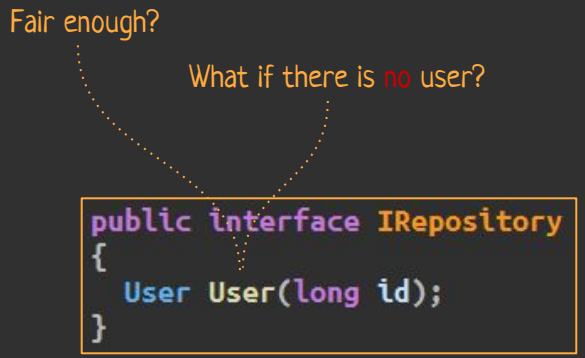
Cool types

Maybe monad

Fair enough?

```
public interface IRepository
{
  User User(long id);
}
```

Cool types


Fair enough?
What if there is no user?
public interface IRepository
{
User User(long id);
}

Cool types

Maybe monad

```
public interface IRepository
{
  UserOrNull User(long id);
}
```

Cool types

Maybe monad

```
public interface IRepository
{
  Maybe<User> User(long id);
}
```

Cool types

Maybe monad

Cool types

Maybe monad

```
public interface IEquipment
{
  Maybe<IWeapon> Weapon();
}
```

Cool types

Maybe monad

```
public interface IBoss
{
  Maybe<Money> Rise();
}
```

Cool types

Cool types

Result monad

Cool types

Result monad

```
public static int Parse(string s)
```

Cool types

Fair enough?

```
public static int Parse(string s)
```

Cool types

Fair enough?

```
public static int Parse(string s)
{
  // ...
  if (StringIsNotANumber)
    throw new Exception("Specified string is not a number!");
  // ...

  return Something;
}
```

Cool types

Result monad

Fair enough?

Breaks flow

```
public static int Parse(string s)
{
  // ...
  if (StringIsNotANumber)
    throw new Exception("Specified string is not a number!");
  // ...

  return Something;
}
```

Cool types

Result monad

Fair enough?

Breaks flow

```
public static Result<int> Parse(string s)
{
  // ...
  if (StringIsNotANumber)
    throw new Exception("Specified string is not a number!");
  // ...

  return Something;
}
```

Cool types

```
public struct Result<T>
{
    private readonly T _value;
    private readonly string _errorMessage;

    public Result(T value){...}
    public Result(string errorMessage){...}

    public T Value
    {
        get
        {
            if (_errorMessage != null)
                throw new Exception(_errorMessage);

            return _value;
        }
    }

    public string Error => _errorMessage;
}
```

Fair enough?

```
static Result<int> Parse(string s)

tringIsNotANumber)
ow new Exception("Specified string is not a number!");

  return Something;
}
```

Cool types

Result monad

Fair enough?

Breaks flow

```
public static Result<int> Parse(string s)
{
  // ...
  if (StringIsNotANumber)
    throw new Exception("Specified string is not a number!");
  // ...

  return Something;
}
```

Cool types

Result monad

Fair enough?

Breaks flow

```
public static Result<int> Parse(string s)
{
  // ...
  if (StringIsNotANumber)
    throw new Exception("Specified string is not a number!");
  // ...

  return Something;
}
```

Cool types

Result monad

Fair enough?

Breaks flow

```
public static Result<int> Parse(string s)
{
  // ...
  if (StringIsNotANumber)
    throw new Exception("Specified string is not a number!");
  // ...

  return Result.Of(Something);
}
```

Cool types

Result monad

Fair enough?

Breaks flow

```
public static Result<int> Parse(string s)
{
  // ...
  if (StringIsNotANumber)
    throw new Exception("Specified string is not a number!");
  // ...

  return Result.Of(Something);
}
```

Cool types

Fair enough!

```
public static Result<int> Parse(string s)
{
  // ...
  if (StringIsNotANumber)
    return Result.Fail<int>("Specified string is not a number!")
  // ...

  return Result.Of(Something);
}
```

Cool types

Result monad

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;



}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)



}
```

Cool types

Result monad

```
public static Result When(bool condition, string error = "")
{
  if (condition)
    return Result.Succeded();
  return Result.Failed(error);
}
```

```
xecute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)



}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)



}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)



}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)


public Maybe<IItem> FindIn(IWorld world) => world.FirstOrDefault<IItem>(x => x.Entity.Id.Value == Id);

}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)



}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)
```

```
public static Result<T> With<T>(this Result self, Func<Maybe<T>> other) =>
    self.Success ? other().AsResult() : Result.Failed<T>(self.Error);
```

```
}
```

Cool types

Result monad

```
public static Result<T> With<T>(this Result self, T other) => self.With(() => other);
public static Result<T> With<T>(this Result self, Func<T> other) =>
    self.Success ? Result.Succeded(other()) : Result.Failed<T>(self.Error);

public static Result<T> With<T>(this Result self, Maybe<T> other) => self.With(() => other);
public static Result<T> With<T>(this Result self, Func<Maybe<T>> other) =>
    self.Success ? other().AsResult() : Result.Failed<T>(self.Error);

public static Result<T> With<T>(this Result self, Result<T> other) => self.With(() => other);
public static Result<T> With<T>(this Result self, Func<Result<T>> other) =>
    self.Success ? other() : Result.Failed<T>(self.Error);

public static Result With(this Result self, Result other) => self.With(() => other);
public static Result With(this Result self, Func<Result> other) =>
    self.Success ? other() : self;
```

```
protected
{
    IItem
    IPlay

    retur
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)
```

```
public static Result<T> With<T>(this Result self, Func<Maybe<T>> other) =>
    self.Success ? other().AsResult() : Result.Failed<T>(self.Error);
}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)



}
```

Cool types

Result monad

```
public static Result OnSuccess(this Result self, Action action)
{
    if (self.Success)
        action();

    return self;
}
```

```
upMessage message)
    IPlayer player = null;

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)


}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)



}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)
        .With(() => _world.Player(message.Name)).OnSuccess(x => player = x)


}
```

Cool types

Result monad

```
public static Maybe<IPlayer> Player(this IWorld self, string name) =>
    self
        .ObjectsOfAspect<IPlayer>()
        .Current
        .FirstOrDefault(x => x.Name == name)
        .Maybe();
```

```
protected override Re
{
    IItem item = null
    IPlayer player = 

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)
        .With(() => _world.Player(message.Name)).OnSuccess(x => player = x)


}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)
        .With(() => _world.Player(message.Name)).OnSuccess(x => player = x)
        .With(() => player.Entity.Pickup(_world, item))


}
```

Cool types

Result monad

```
protected override
{
    IItem item = nu
    IPlayer player

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)
        .With(() => _world.Player(message.Name)).OnSuccess(x => player = x)
        .With(() => player.Entity.Pickup(_world, item))


}
```

```
public static Result Pickup(this IEntity self, IWorld world, IItem item) =>
    Result
        .Of(() => self.Aspect<IEquipment>().Bags().FirstOrDefault())
        .With(x => x.Put(item))
        .With(() => world.Remove(item.Entity));
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)
        .With(() => _world.Player(message.Name)).OnSuccess(x => player = x)
        .With(() => player.Entity.Pickup(_world, item))


}
```

Cool types

Result monad

```
protected override Result<PickupResultMessage> Execute(PickupMessage message)
{
    IItem item = null;
    IPlayer player = null;

    return Result
        .When(message.Name != null)
        .With(() => message.Item.FindIn(_world)).OnSuccess(x => item = x)
        .With(() => _world.Player(message.Name)).OnSuccess(x => player = x)
        .With(() => player.Entity.Pickup(_world, item))
        .As(PickupResultMessage.Create)
        .AsResult();
}
```

Cool types

Cool types

 Fair types and signatures

Cool types

- ♥ Fair types and signatures
- ♥ Fluent thinking

Cool types

- ♥ Fair types and signatures
- ♥ Fluent thinking
- ♥ Declarative expressions

Cool types

Последний раздел

Я хотел назвать его “Диалектический эпилог”, но придумал это позже и забыл исправить

Последний раздел

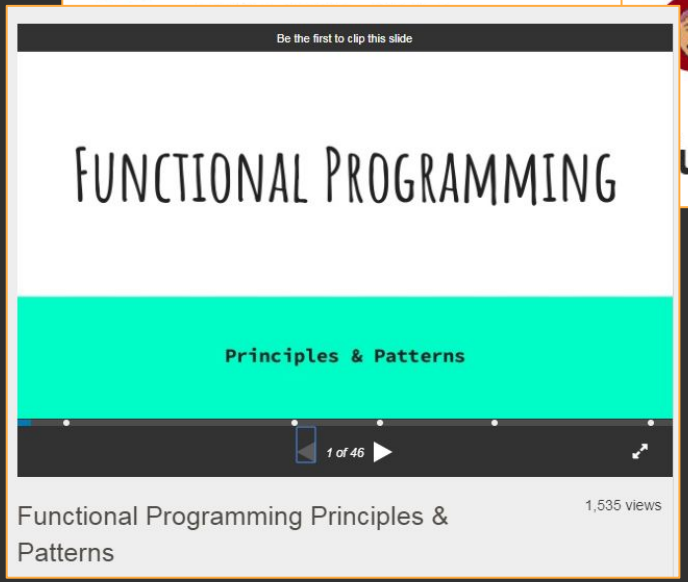
FUNCTIONAL PROGRAMMING PRINCIPLES EVERY IMPERATIVE PROGRAMMER SHOULD USE

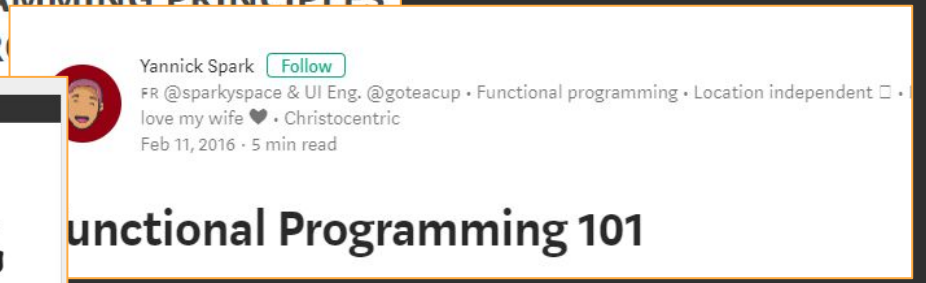
FUNCTIONAL PROGRAMMING PRINCIPLES
EVERY IMPERATIVE PR
SHOULD USE

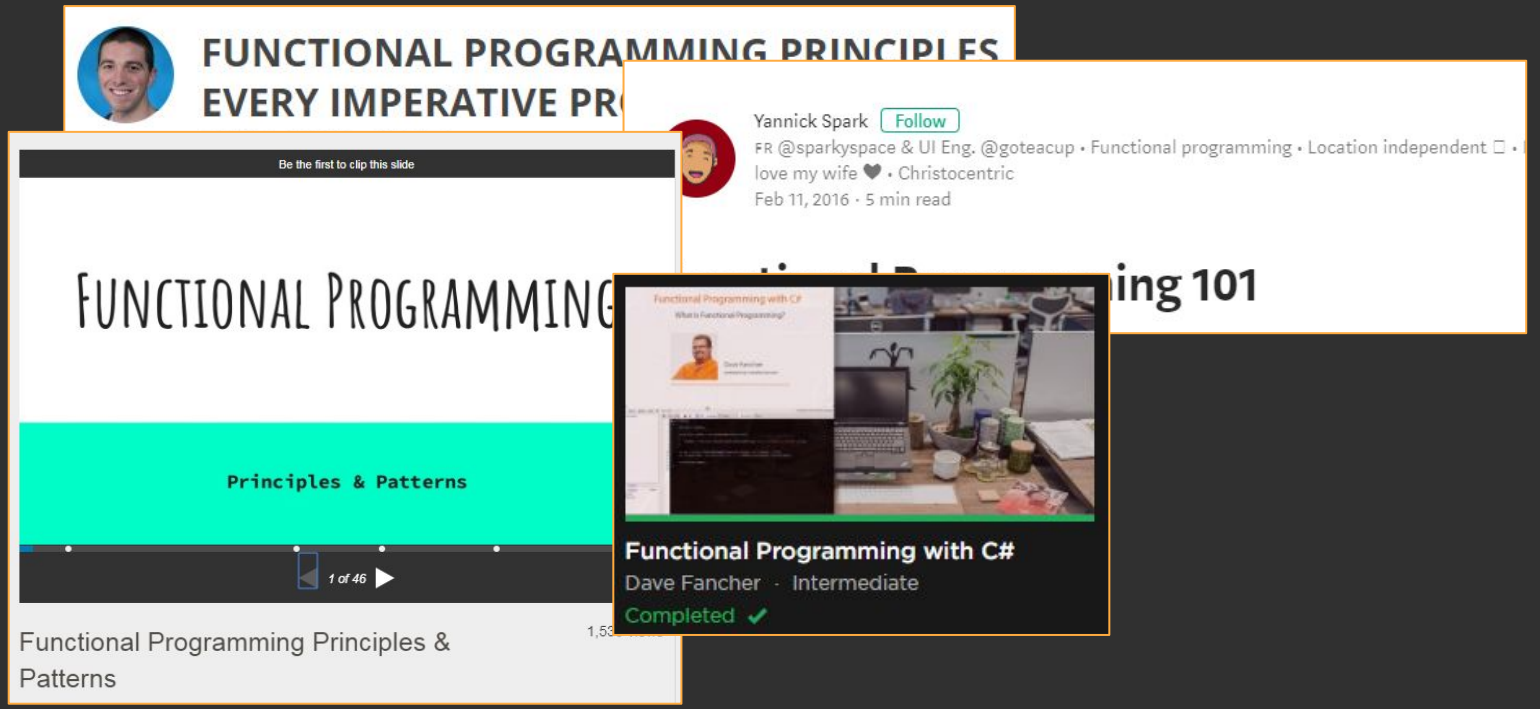
Yannick Spark Follow
FR @sparkyspace & UI Eng. @goteacup • Functional programming • Location independent □ •
love my wife ♥ • Christocentric
Feb 11, 2016 · 5 min read

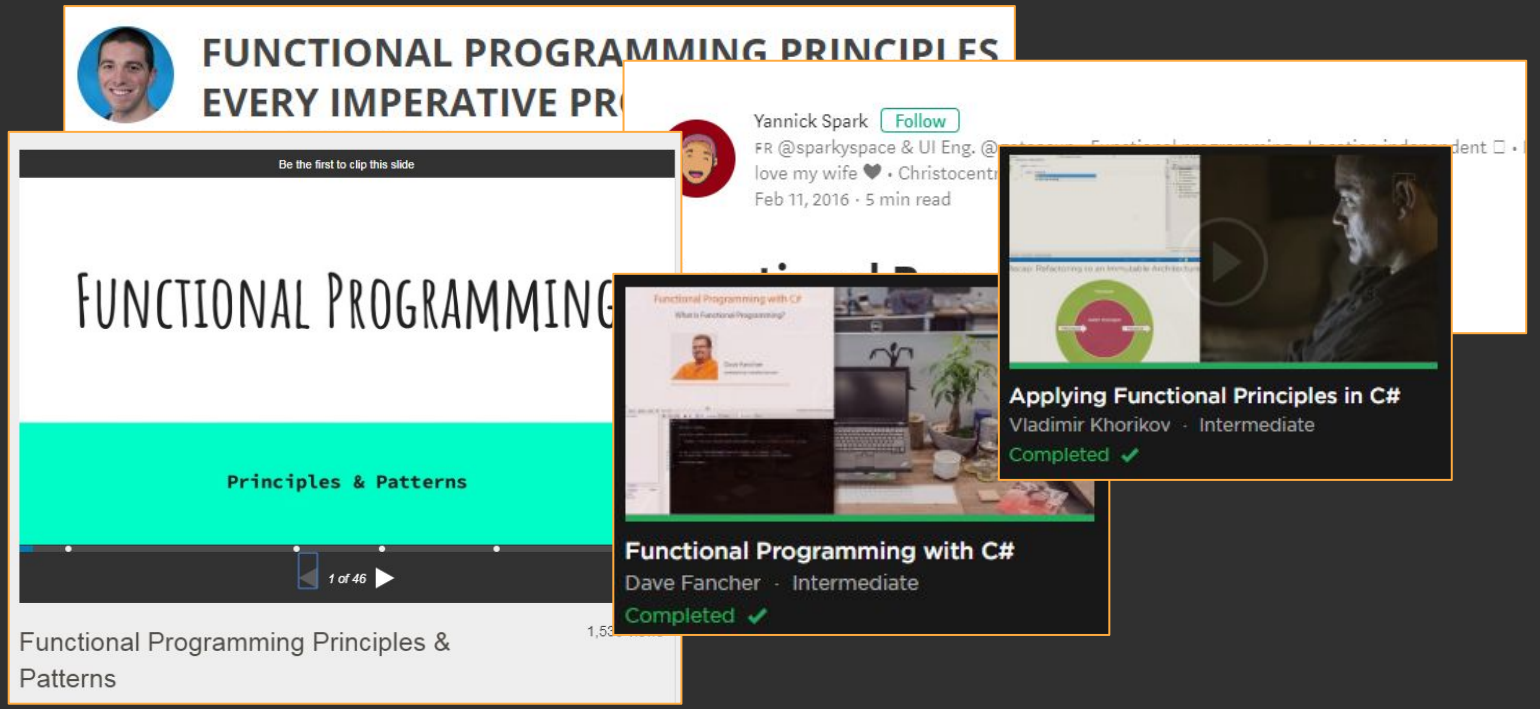
Functional Programming 101


FUNCTIONAL PROGRAMMING PRINCIPLES
EVERY IMPERATIVE PR

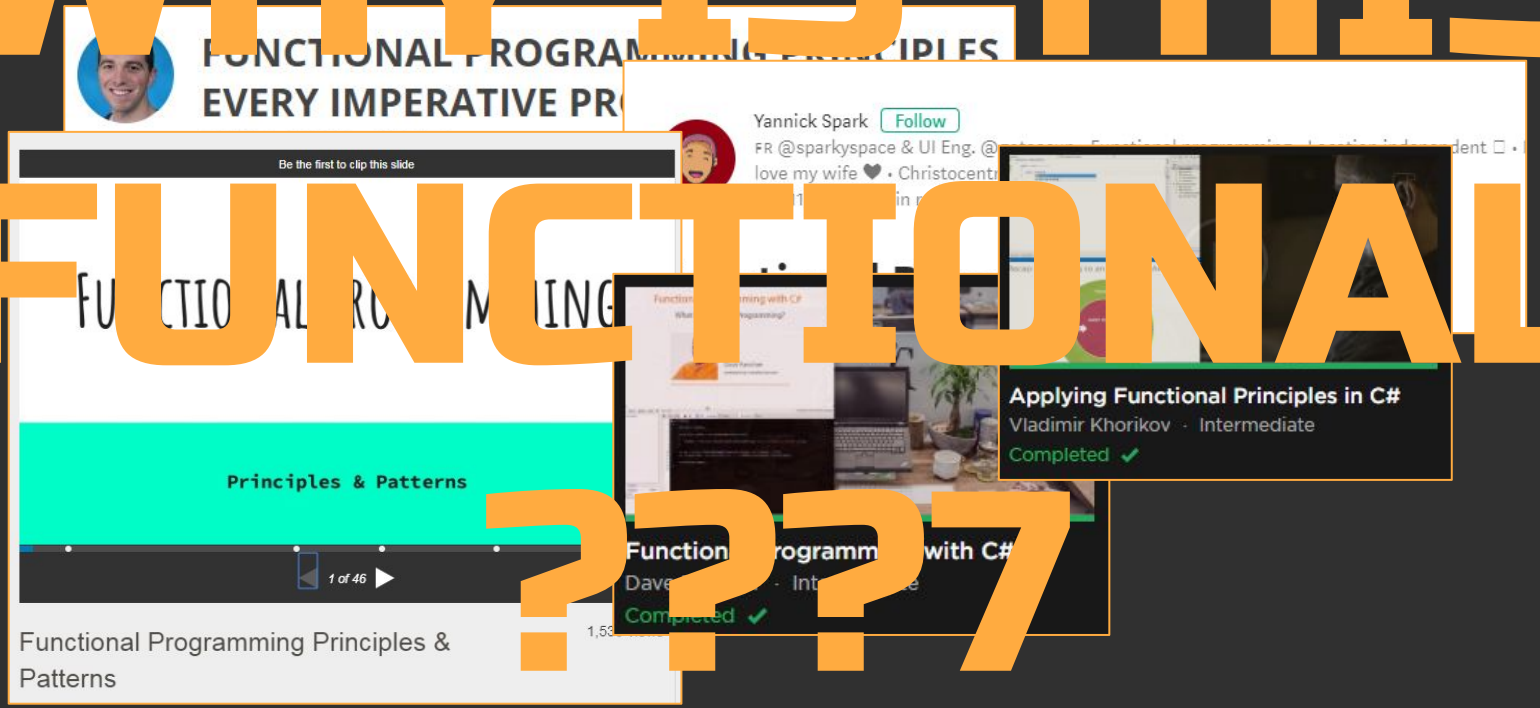

Be the first to clip this slide
FUNCTIONAL PROGRAMMING
Principles & Patterns
1 of 46
Functional Programming Principles & Patterns
1,535 views

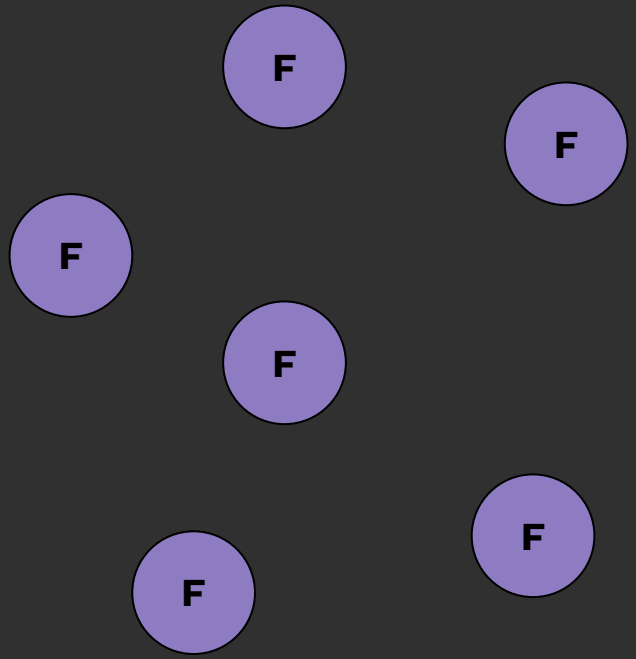

Yannick Spark
Follow
FR @sparkyspace & UI Eng. @goteacup • Functional programming • Location independent
love my wife ♥ • Christocentric
Feb 11, 2016 · 5 min read
unctional Programming 101

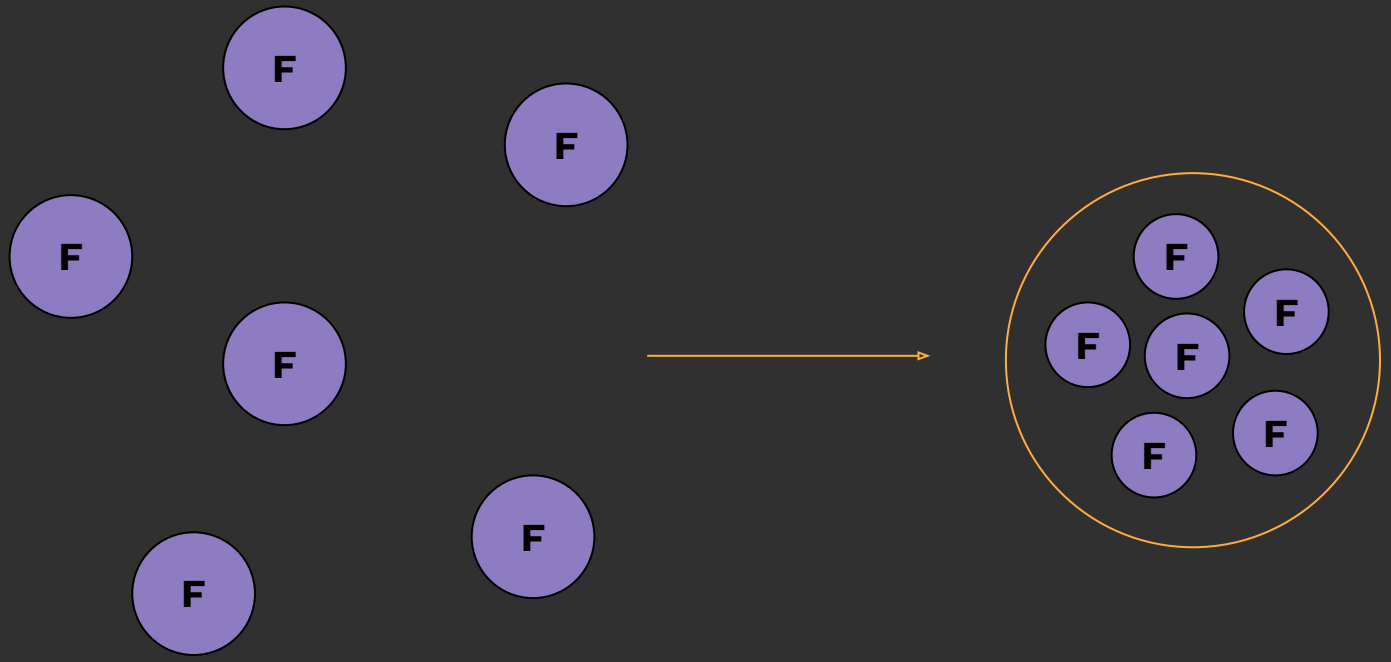

FUNCTIONAL PROGRAMMING PRINCIPLES
EVERY IMPERATIVE PR
Yannick Spark
Follow
FR @sparkyspace & UI Eng. @goteacup • Functional programming • Location independent
love my wife ♥ • Christocentric
Feb 11, 2016 · 5 min read
ing 101
Be the first to clip this slide
FUNCTIONAL PROGRAMMING
Principles & Patterns
1 of 46
Functional Programming Principles & Patterns
Functional Programming with C#
Dave Fancher · Intermediate
Completed

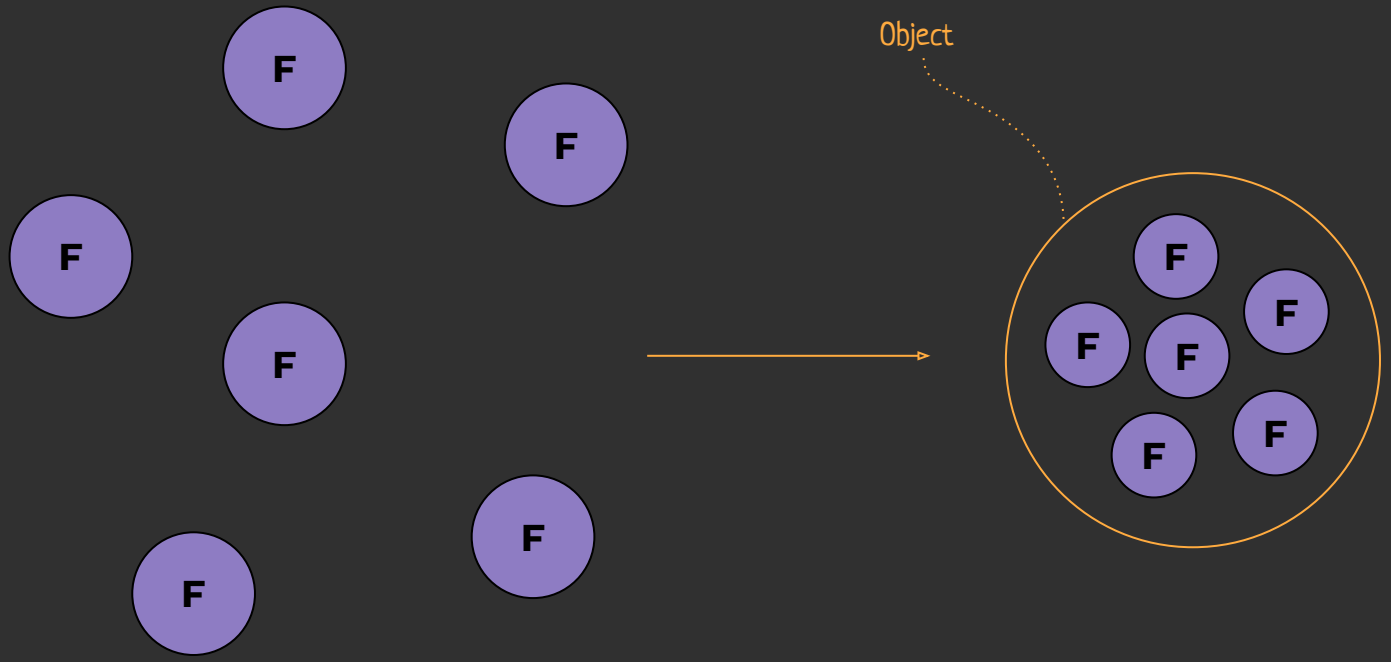

FUNCTIONAL PROGRAMMING PRINCIPLES
EVERY IMPERATIVE PR
Yannick Spark
Follow
Feb 11, 2016 · 5 min read
Be the first to clip this slide
FUNCTIONAL PROGRAMMING
Principles & Patterns
1 of 46
Functional Programming Principles & Patterns
Applying Functional Principles in C#
Vladimir Khorikov · Intermediate
Completed
Functional Programming with C#
Dave Fancher · Intermediate
Completed

WHY IS THIS FUNCTIONAL ????




F
F
F
F
F
F


F
F
F
F
F
F
F
F
F
F
F
F
F


F
F
F
F
F
F
Object
F
F
F
F
F
F

```
public struct Point : IShape, IEquatable<Point>
{
    public static readonly Point Zero = new Point();

    public Point(int x, int y)
    {
        X = x;
        Y = y;
    }

    public int X { get; }
    public int Y { get; }

    public Point WithX(int x) => new Point(x, Y);
    public Point WithY(int y) => new Point(X, y);

    Other
}
```

Pure object?

```
public struct Point : IShape, IEquatable<Point>
{
    public static readonly Point Zero = new Point();

    public Point(int x, int y)
    {
        X = x;
        Y = y;
    }

    public int X { get; }
    public int Y { get; }

    public Point WithX(int x) => new Point(x, Y);
    public Point WithY(int y) => new Point(X, y);

    Other
}
```

```
public struct Nullable<T> where T : struct
```

```
public struct Nullable<T> where T : struct
```

```
public interface IRepository
{
  Maybe<User> User(long id);
}
```

```
public struct Nullable<T> where T : struct
```

```
public interface IEquipment
{
  Maybe<IWeapon> Weapon();
}
```

```
public interface IRepository
{
  Maybe<User> User(long id);
}
```

```
public struct Nullable<T> where T : struct
```

```
public interface IEquipment
{
  Maybe<IWeapon> Weapon();
}
```

```
public interface IRepository
{
  Maybe<User> User(long id);
}
```

```
public static int Divide(int a, NonZeroInt b) => a / b;
```

```
public struct Nullable<T> where T : struct
```

```
public interface IEquipment
{
  Maybe<IWeapon> Weapon();
}
```

```
public interface IRepository
{
  Maybe<User> User(long id);
}
```

```
public static int Divide(int a, NonZeroInt b) => a / b;
```

Functional programming

```
public struct Nullable<T> where T : struct
```

```
public interface IEquipment
{
  Maybe<IWeapon> Weapon();
}
```

```
public interface IRepository
{
  Maybe<User> User(long id);
}
```

```
public static int Divide(int a, NonZeroInt b) => a / b;
```

~~Functional~~ Normalniy programming

```
var world = new DefaultWorld();
var player = world.GetPlayerByName("Jerome Salinger");
```

```
var world = new DefaultWorld();
var player = world.Player("Jerome Salinger");
```

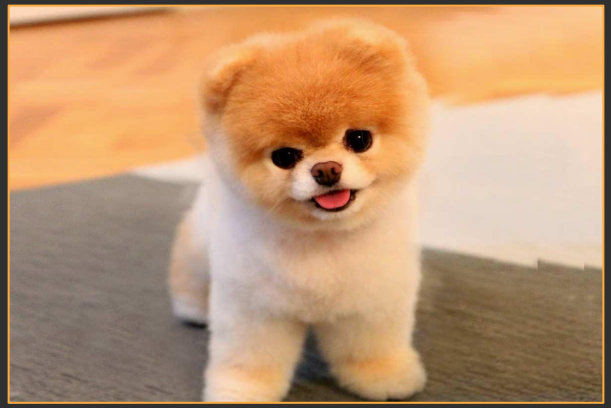
```
public interface IEnumerable<T>
{
  IEnumerator<T> GetEnumerator();
}
```

```
public interface ISequence<T>
{
  ICursor<T> Cursor();
}
```

```
public static class ListSorter
{
  public static List<T> QuickSort<T>(List<T> source){...}
}
```

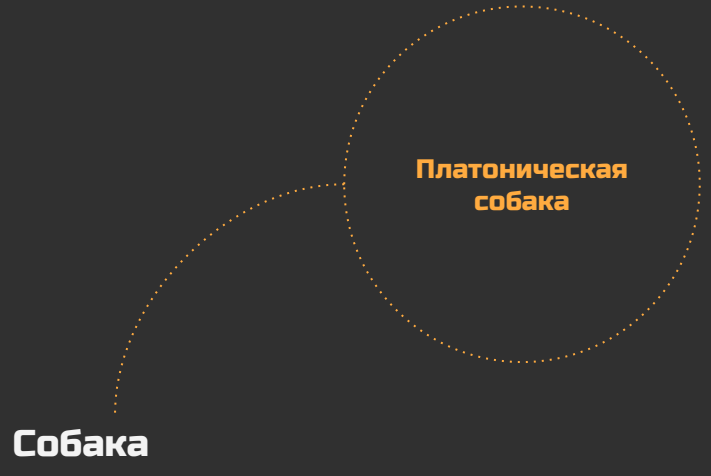
```
public class SortedList<T> : List<T>
{
  // Some lazy sort implementation, for example.
}
```

Собака



Собака

Собака


Платоническая собака
Собака

Собака

Собака
Sobaka

Собака
Sobaka
Dog

Синтаксис

Собака
Sobaka
Dog

Синтаксис

Собака
Sobaka
Dog

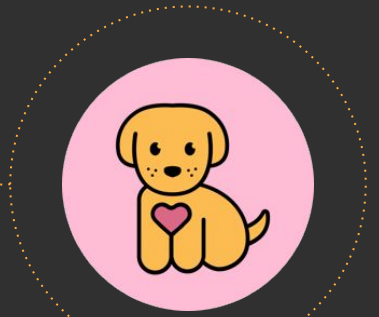
Семантика

Форма

Собака
Sobaka
Dog

Семантика

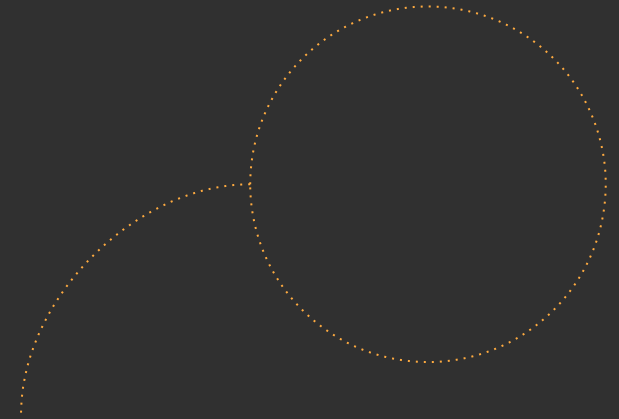
Форма



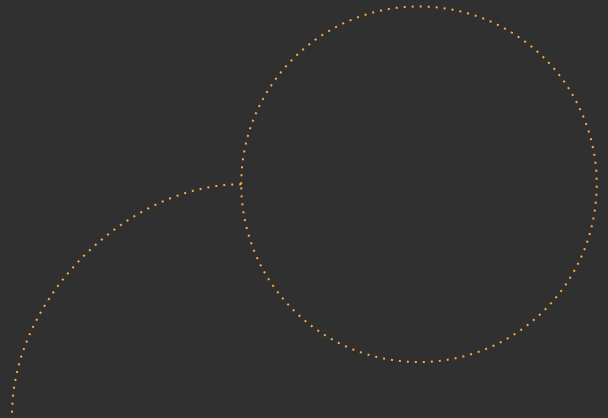
Собака
Sobaka
Dog

Содержание

Собака
Sobaka
Dog



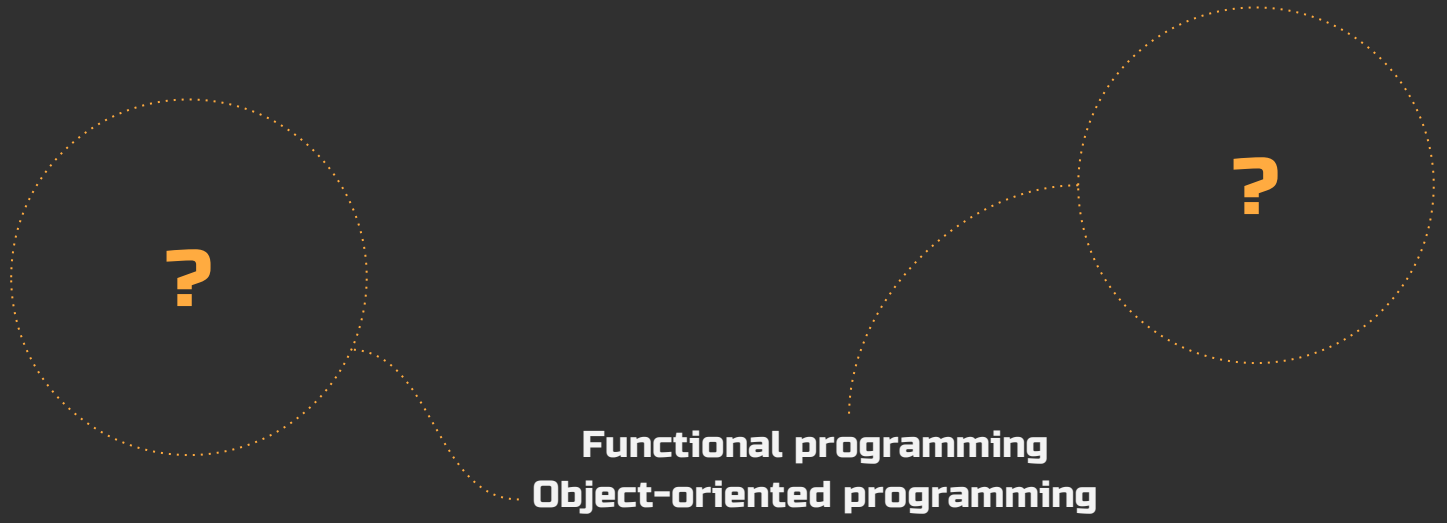
Собака
Sobaka
Dog

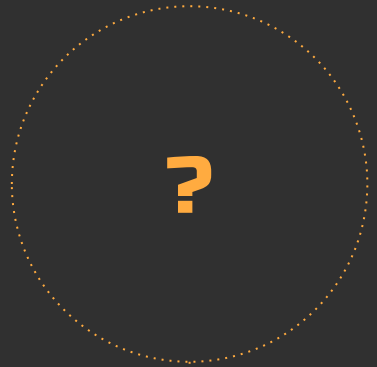


Functional programming

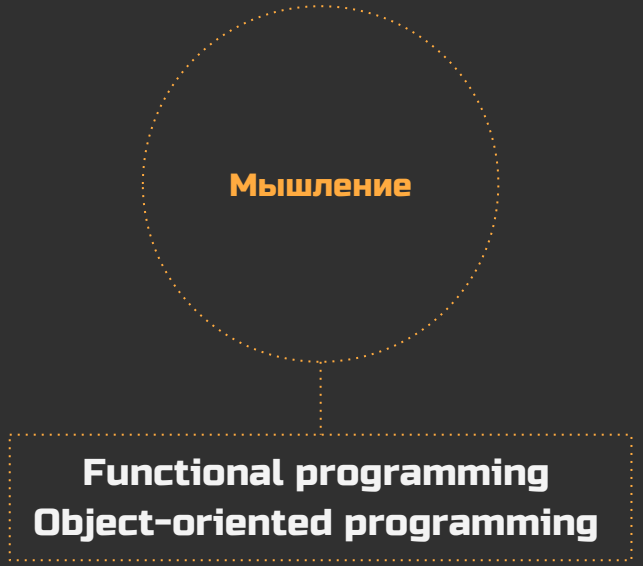
Functional programming

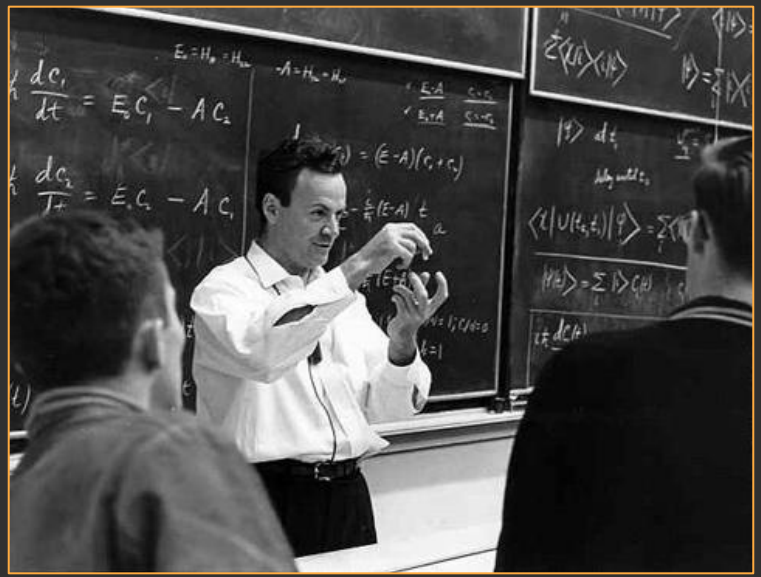
Functional programming
Object-oriented programming


?
?
Functional programming
Object-oriented programming

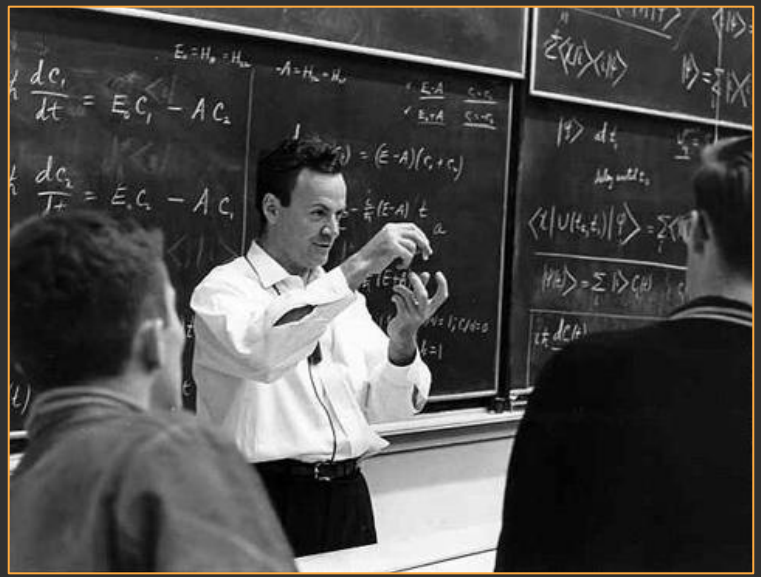


Functional programming
Object-oriented programming

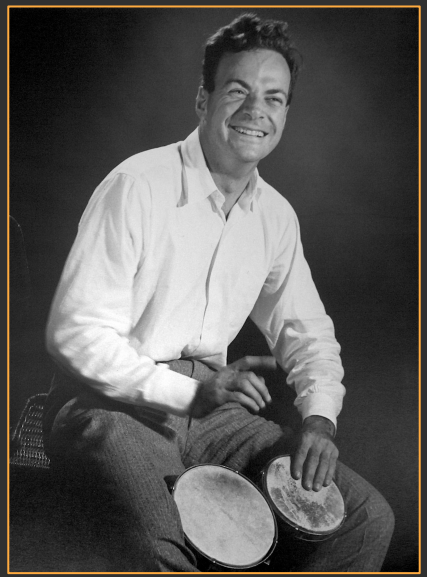

Мышление
Functional programming
Object-oriented programming


$\frac{dC_1}{dt} = E_2C_1 - AC_2$
$\frac{dC_2}{dt} = E_1C_2 - AC_1$

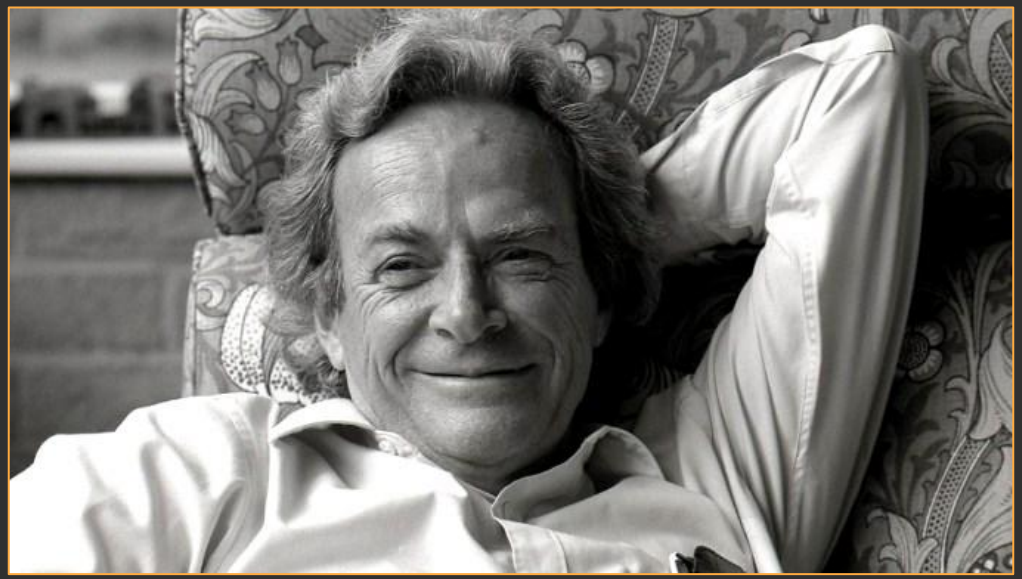
Richard Feynman (1918 – 1988)

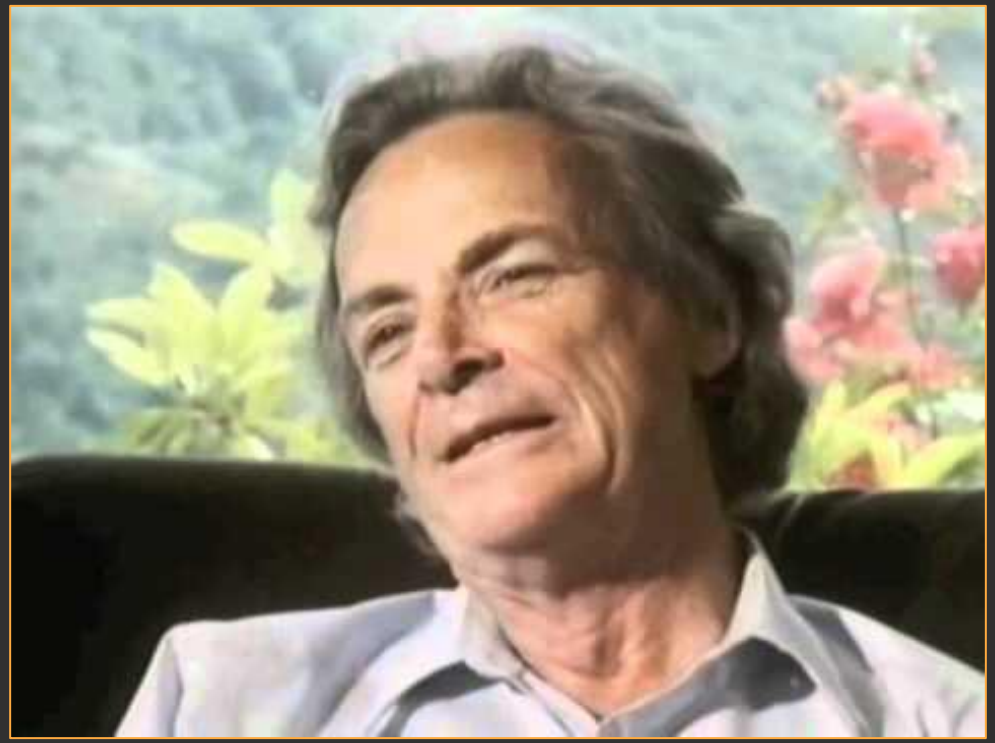


Richard Feynman (1918 – 1988)



Richard Feynman (1918 – 1988)





❤️ Names don’t constitute knowledge

❤️ Names don’t constitute knowledge

❤️ So maybe functional or object-oriented or whatever there is — just a bunch of names

❤️ Names don’t constitute knowledge

❤️ So maybe functional or object-oriented or whatever there is — just a bunch of names of one thing?

The End.