История языков программирования

Для тех, кто пишет на C#

Марк Шевченко

Московский клуб программистов

https://prog.msk.ru

https://markshevchenko.pro

@markshevchenko

Одна простая задача

```
int a = 2;
int b = 2 * ++a + 3 * ++a;
```

Одна простая задача

```
int a = 2;
int b = 2 * ++a + 3 * ++a;

//  C#: 18 потому, что 2 * 3 + 3 * 4
```

Одна простая задача

```
int a = 2;
int b = 2 * ++a + 3 * ++a;

//  C#: 18 потому, что 2 * 3 + 3 * 4
// C++: 20 потому, что?
```

Порядок вычисления операндов

```
a + b + c
```

Порядок вычисления операндов

`a + b + c` означает `(a + b) + c`

`a + b`

Порядок вычисления операндов

```
a + b + c означает (a + b) + c
a + b
a1 * a2 + b1 * b2
```

Дело в другом

```
int a = 2;
int b = 2 * ++a + 3 * ++a;

// 2 * 3 + 3 * 4 == 18
// 2 * 4 + 3 * 3 == 17
```

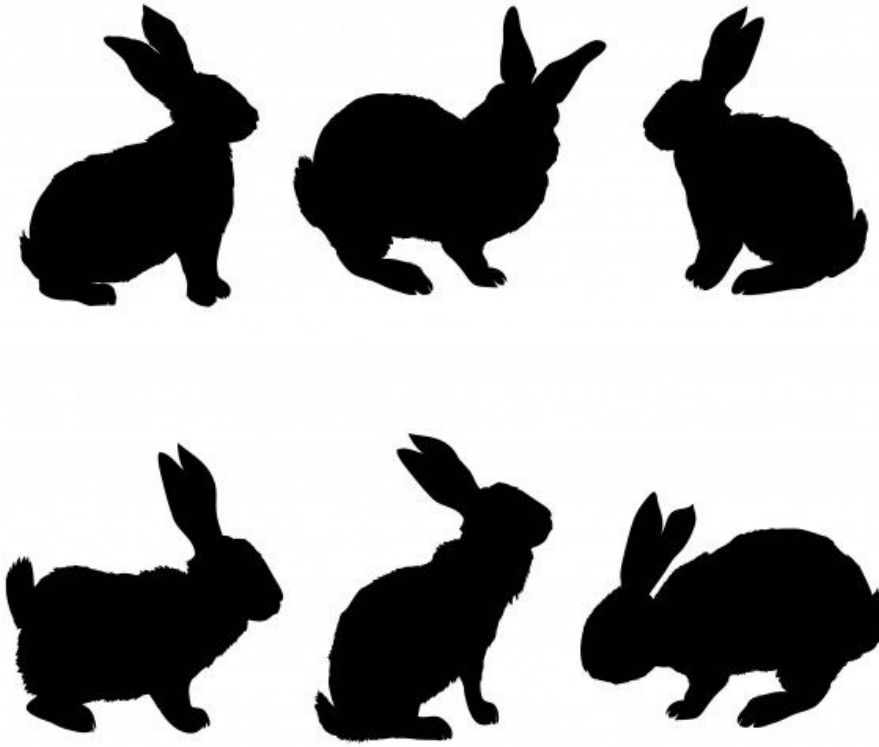
Дело в другом

```
int a = 2;
int b = 2 * ++a + 3 * ++a;

int b = (2 + 3) * (++ ++a);
```

В C++ скорость важнее безопасности, а в C# — наоборот

Задача



0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
55, 89, 144, 233, 377, 610,
987, 1597, 2584, 4181...

Язык ассемблера

```
@format = private constant [3 x i8] c"%d\0A"
declare i32 @printf(i8*, ...)

define i32 @main() {
entry:
  %a = alloca i32, align 4
  store i32 0, i32* %a
  %b = alloca i32, align 4
  store i32 1, i32* %b

  %i = alloca i32, align 4
  store i32 0, i32* %i

  br label %l1
l1:
```

Язык ассемблера

```
  %0 = load i32* %a, align 4
  %1 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds
            ([3 x i8]* @format, i32 0, i32 0), i32 %0)
  %2 = load i32* %b, align 4
  %3 = add i32 %0, %2
  store i32 %3, i32* %b
  store i32 %2, i32* %a

  %4 = load i32* %i, align 4
  %5 = add i32 %4, 1
  store i32 %5, i32* %i

  %6 = icmp eq i32 %5, 20
  br i1 %6, label %l2, label %l1

l2:
  ret i32 0
}
```

Fortran

```
program fibonacci
    integer a, b, t
    a = 0
    b = 1
    i = 1

10  print *, a
    t = b
    b = a + b
    a = t

    i = i + 1
    if (i .le. 20) goto 10
end program fibonacci
```

Go To Statement Considered Harmful

Структурное программирование позволило компилятору отслеживать поток управления.

LISP

```
(defun fibonacci (n)
    (defun build-fibonacci (i a b result)
        (cond
            ((= i 0) result)
            (t (build-fibonacci (- i 1) b (+ a b) (cons a result)))
        )
    )

    (build-fibonacci n 0 1 '())
)
```

Pascal

```
program Fibonacci;

var
  A, B, T: Integer;
  I: Integer;

begin
  A := 0;
  B := 1;

  for I := 1 to 20 do
  begin
    WriteLn(A);
    T := B;
    B := A + B;
    A := T;
  end;
end.
```

Pascal/C

```pascal
program Fibonacci;

var
  A, B, T: Integer;
  I: Integer;

begin
  A := 0;
  B := 1;

  for I := 1 to 20 do
  begin
    WriteLn(A);
    T := B;
    B := A + B;
    A := T;
  end;
end.
```

```c
#include <stdio.h>

void main()
{
    int a = 0, b = 1, t;

    for (int i = 0; i < 20; i++)
    {
        printf("%d\n", a);

        t = b;
        b = a + b;
        a = t;
    }
}
```

Pascal/C

Python

```
a, b = 0, 1
for i in range(20):
    print(a)
    a, b = b, a + b
```

Prolog

```
fibonacci(1, [0]).
fibonacci(2, [1,0]).
fibonacci(N, [R,A,B|Cs]) :-
    N > 2,
    N1 is N - 1,
    fibonacci(N1,[A,B|Cs]),
    R is A + B.
```

```
?- fibonacci(20, X).
X = [4181, 2584, 1597, 987, 610, 377, 233, 144, 89|...] ;
false.
```

SQL

```
WITH Fibonacci AS
(
  SELECT 1 AS Number, 0 AS A, 1 AS B
   UNION ALL
  SELECT Number + 1 AS Number, B AS A, (A + B) AS B
    FROM Fibonacci
   WHERE Number < 20
)
SELECT A FROM Fibonacci
```

Полиморфизм

```
a + b
```

Полиморфизм

```
a + b

f(a, b)
```

Полиморфизм

```
a + b

f(a, b)

a.f(b)
```

Ad hoc полиморфизм (overload)

```
static byte[] GetBytes(uint x)
{
    var result = new byte[4];
    result[0] = (byte)(x & 0xFFu);
    result[1] = (byte)(x >> 8 & 0xFFu);
    result[2] = (byte)(x >> 16 & 0xFFu);
    result[3] = (byte)(x >> 24 & 0xFFu);

    return result;
}
```

```
static byte[] GetBytes(ulong x)
{
    var result = new byte[8];
    result[0] = (byte)(x & 0xFFul);
    result[1] = (byte)(x >> 8 & 0xFFul);
    result[2] = (byte)(x >> 16 & 0xFFul);
    result[3] = (byte)(x >> 24 & 0xFFul);
    result[4] = (byte)(x >> 32 & 0xFFul);
    result[5] = (byte)(x >> 40 & 0xFFul);
    result[6] = (byte)(x >> 48 & 0xFFul);
    result[7] = (byte)(x >> 56 & 0xFFul);

    return result;
}
```

Ad hoc полиморфизм (override)

```
class Rectangle : Shape
{
      public double Width { get; }
      public double Height { get; }

      public Rectangle(double width,
                       double height)
      {
            Width = width;
            Height = height;
      }

      public override double GetArea()
      {
            return width * height;
      }
}
```

```
class Circle : Shape
{
      public double Radius { get; }

      public Circle(double radius)
      {
            Radius = radius;
      }

      public override double GetArea()
      {
            return Math.PI * radius
                           * radius;
      }
}
```

Параметрический полиморфизм

```
type
  IntegerBinaryNode = record
    Value: Integer;
    Left: ^IntegerBinaryNode;
    Right: ^IntegerBinaryNode;
  end;

  StringBinaryNode = record
    Value: String;
    Left: ^StringBinaryNode;
    Right: ^StringBinaryNode;
  end;
```

Параметрический полиморфизм

```
type 'a binary_tree = Leaf
                    | Node of 'a * 'a binary_tree * 'a binary_tree


let rec contains x t = match t with
                     | Leaf -> false
                     | Node (y, left, right) -> x = y
                                             || contains x left
                                             || contains x right
```

ML

```
let rec fibonacci n =
  match n with
  | 1 -> [0]
  | 2 -> [1; 0]
  | _ -> match fibonacci (n - 1) with
         | a::b::cs -> (a + b)::a::b::cs
         | _        -> []

printfn "%A" (fibonacci 20)
```

SmallTalk

```
a := 0.
b := 1.
20 timesRepeat: [
   a displayNl.
   t := b.
   b := a + b.
   a := t.
].
```

C++

C++

```
template<int n> class Fibonacci {
public:
	static int get() {
		return Fibonacci<n - 1>::get() + Fibonacci<n - 2>::get();
	}
};

template<> class Fibonacci<0> {
public:
	static int get() {
		return 0;
	}
};

template<> class Fibonacci<1> {
public:
	static int get() {
		return 1;
	}
};

std::cout << Fibonacci<20>::get() << std::endl;
```

Множественное наследование

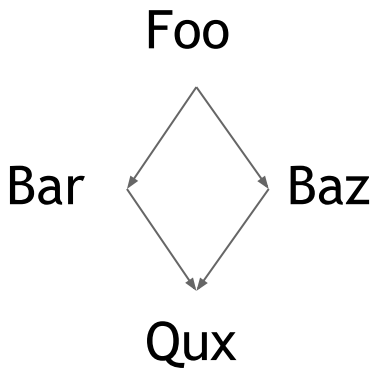
```
class Foo {
public:
      int foo;

      virtual void print() { std::cout << "Foo" << std::endl; }
      virtual int pure() = 0;
};

class Bar : Foo {
public:
      virtual void print() { foo = 1;  std::cout << "Bar" << std::endl; }
};

class Baz : Foo {
public:
      virtual void print() { foo = 2;  std::cout << "Baz" << std::endl; }
};

class Qux : Bar, Baz {
public:
      virtual int pure() { return 1; }
};
```



Множественное наследование

```
interface IFoo
{
    string Bar { get; set; }

    event EventHandler Baz;
}
```

Java

```java
public class Fibonacci {
  public static void main(String []args) {
    int[] fibonacci = new int[20];
    fibonacci[0] = 0;
    fibonacci[1] = 1;

    for (int i = 2; i < fibonacci.length; i++)
      fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2];

    System.out.println(java.util.Arrays.toString(fibonacci));
  }
}
```

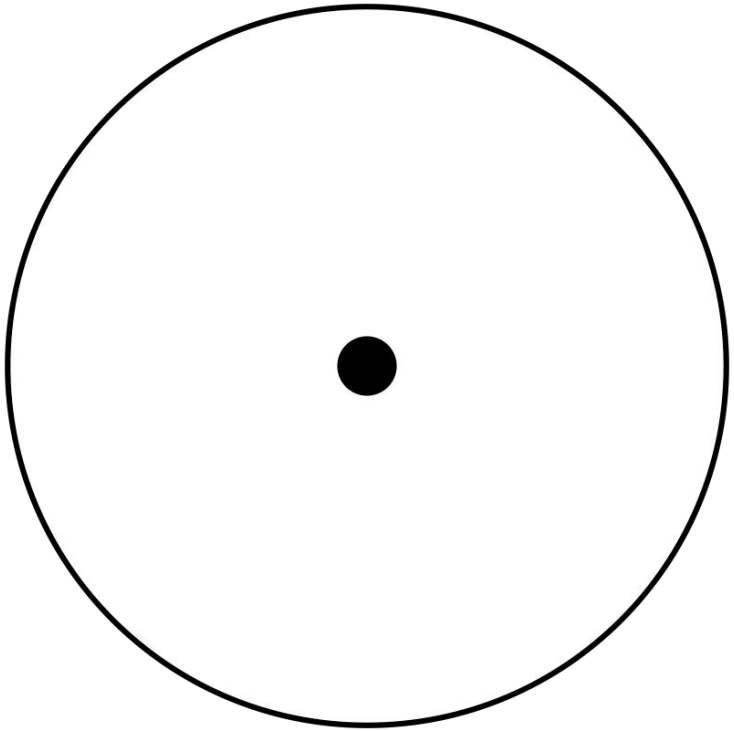
Haskell

```haskell
module Main where

fibonacci = 0 : 1 : zipWith (+) fibonacci (tail fibonacci)

main = print $ take 20 fibonacci
```

async/await



async/await

```
var result = ((int?)3).Bind(x => (int?)(x * x))
                      .Bind(x => (double?)(Math.PI * x))
                      .Bind(x => (double?)(2 * x));
```

```
public static TResult? Bind<TSource, TResult>(this TSource? source,
                                              Func<TSource, TResult?> action)
    where TSource : struct
    where TResult : struct
{
    if (source.HasValue)
        return action(source.Value);
    else
        return null;
}
```

async/await

```
var result = ((int?)3).Bind(x => (int?)(x * x))
                      .Bind(x => (double?)(Math.PI * x))
                      .Bind(x => (double?)(2 * x));
```

```
var result = Task.FromResult(3).ContinueWith(t => t.Result * t.Result)
                               .ContinueWith(t => Math.PI * t.Result)
                               .ContinueWith(t => 2 * t.Result);
```

async/await

```
var result = ((int?)3).Bind(x => (int?)(x * x))
                      .Bind(x => (double?)(Math.PI * x))
                      .Bind(x => (double?)(2 * x));
```

```
result = Just 3 >>= \x -> Just (x * x)
                >>= \x -> Just (pi * fromIntegral x)
                >>= \x -> Just (2.0 * x)
```

async/await

```
var result = ((int?)3).Bind(x => (int?)(x * x))
                      .Bind(x => (double?)(Math.PI * x))
                      .Bind(x => (double?)(2 * x));
```

```
result = Just 3 >>= \x -> Just (x * x)
                >>= \x -> Just (pi * fromIntegral x)
                >>= \x -> Just (2.0 * x)
```

```
result = do
         x <- Just 3
         y <- Just (x * x)
         z <- Just (pi * fromIntegral y)
         Just (2.0 * z)
```

async/await

```
do a1 <- async (getURL url1)
   a2 <- async (getURL url2)
   page1 <- wait a1
   page2 <- wait a2
   ...
```

async/await

```
async {
    let uri = new System.Uri(url)
    let client = new WebClient()
    let! html = client.AsyncDownloadString(uri)
    printfn "%s" html
}
```

C#

```
static IEnumerable<int> Fibonacci()
{
    int a = 0;
    int b = 1;

    while (b <= int.MaxValue - a)
    {
        yield return a;
        int t = b;
        b = a + b;
        a = t;
    }

    yield return a;
    yield return b;
}
```

Ссылки

- Миран Липовача (про монады главы 11-13)
  http://learnyouahaskell.com/chapters
- Виталий Брагилевский про монады
  https://youtu.be/IkXg_mjNgG4
- Koen Claessen, A Poor Man's Concurrency Monad
  http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.8039&rep=rep1&type=pdf
- Don Syme, F# Asynchronous Workflows
  https://docs.microsoft.com/ru-ru/archive/blogs/dsyme/introducing-f-asynchronous-workflows

История языков для тех, кто пишет на C#

https://prog.msk.ru

https://markshevchenko.pro

@markshevchenko

- Порядок вычисления операндов
- Числа Фибоначчи
- Язык Ассемблера
- Fortran
- Структурное программирование
- LISP
- Pascal, C
- Андерс Хейлсберг
- Python
- Prolog
- SQL
- Полиморфизм
- ML
- SmallTalk
- C++, шаблоны, наследование
- Java
- Haskell
- async/await