UDC 004.4

EFFICIENCY OF USING CRYSTAL PROGRAMMING LANGUAGE FOR APPLICATION IMPLEMENTATION

O. MIKHNOVICH, O. MIKHNOVICH
Polotsk State University, Belarus

This article discusses the capabilities of the Crystal programming language and its advantages over the C and Ruby programming languages using the implementation of the Tower of Hanoi problem as an example.

Nowadays, there are many different programming languages. Some of them allow developers to write well-readable and efficient codes, while others allow you to focus on the productivity of the future product. Programming languages by type-checking are usually divided into two categories. They are static and dynamic. Static examples are C, Java, C#. Dynamic examples are Python, JavaScript, Ruby. Ruby is one of the popular dynamic interpreted programming languages.

Ruby has an operating system-independent multithreading implementation, strong dynamic typing, garbage collection, and many other features. In terms of syntax, it is close to the Perl and Eiffel languages, in the object-oriented approach to Smalltalk. Also, some features of the language are taken from Python, Lisp, Dylan, and Clu. Nowadays, Ruby is used with the Rails framework. However, Ruby has its drawbacks. The most significant drawback of Ruby is its performance. Consider Crystal programming language as a more productive alternative to Ruby [1].

Crystal is a general-purpose, object-oriented programming language, designed and developed by Ary Borenszweig, Juan Wajnerman. Crystal is statically typed and has Ruby-like syntax. The first official release of the language took place in June 2014. The language compiler was originally written in Ruby until it was rewritten in Crystal in 2013. The language is under active development. Despite the similarities in syntax, Crystal is much more efficient than Ruby to compile into machine code using LLVM, while sacrificing dynamic aspects of the language. According to the test results, Crystal shows similar performance with the C language. The language uses the Boehm garbage collector, has a macro system, it supports generics, as well as overloading methods and operators [2].

Crystal is a programming language with the following goals:

1. Have a syntax like Ruby (but compatibility with it is not a goal).
2. Be statically type-checked, but without having to specify the type of variables or method arguments.
3. Be able to call C code by writing bindings to it in Crystal.
4. Have compile-time evaluation and generation of code, to avoid boilerplate code.
5. Compile to efficient native code.

To compare Crystal performance, we take two programming languages: Ruby and C. The choice fell on these languages since Crystal is positioned as a technology balanced between these languages. To compare the performance, we implement the solution of the Tower of Hanoi problem.

The Tower of Hanoi [3] (also called the Tower of Brahma or Lucas' Tower and sometimes pluralized as Towers) is a mathematical game or a puzzle. It consists of three rods and a few disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape.

There are several approaches to the solution (recursively, a "triangular" solution, a cyclic solution). All of them give identical results. We implement a recursive solution.

The implementation of the solution in Ruby is presented in the listing below.

```
require 'benchmark'
N=20
A=Array.new
B=Array.new
C=Array.new

LOG = false
```

```
def move(n, source, target, auxiliary)
  if n > 0
    move(n - 1, source, auxiliary, target)
    target.append(source.pop)
    print "********\n#{A}\n#{B}\n#{C}\n" if LOG
    move(n - 1, auxiliary, target, source)
  end
end

def run()
  (1..N).each { |v| A << v }
  A.reverse
  move(N, A, C, B)
end

puts "\nRuby work time: #{Benchmark.measure { run() } }"
```

The implementation of the solution in Crystal is presented in the listing below.

```
require "benchmark"
N = 20
A = Array(Int32).new
B = Array(Int32).new
C = Array(Int32).new

LOG = false

def move(n, source, target, auxiliary)
  if n > 0
    move(n - 1, source, auxiliary, target)
    target << source.pop
    print "********\n#{A}\n#{B}\n#{C}\n" if LOG
    move(n - 1, auxiliary, target, source)
  end
end

def run()
  (1..N).each { |v| A << v }
  A.reverse
  move(N, A, C, B)
end

puts "\nCrystal work time: #{Benchmark.measure { run() } }"
```

Listings with solutions to the Tower of Hanoi problem in C, Crystal and Ruby are available on GitHub [4].

After starting the solutions in the selected languages, we obtain the following results (in seconds) (table 1).

Table 1. –Performance Tests (Tower of Hanoi Problem)

Language	Test 1	Test 2	Test 3	Average values
C	0.000895	0.000957	0.000955	0.000936
Crystal	0.021738	0.021611	0.013851	0.019067
Ruby	0.364303	0.359521	0.359140	0.360988

After analyzing the results, we can conclude that Crystal programming language combines, as in C, performance, and readablility, and conciseness; as in Ruby, its syntax and can be used to develop software products

for operating systems of the Unix family. As far as the minuses of this language go, they are a weak community and a small number of libraries so far.

REFERENCES

1. Ruby [Electronic resource] – Mode of access: https://www.ruby-lang.org/. – Date of access: 20.01.2020.
2. Crystal [Electronic resource] – Mode of access: https://crystal-lang.org/. – Date of access: 20.01.2020.
3. The Tower of Hanoi [Electronic resource] – Mode of access: https://en.wikipedia.org/wiki/Tower_of_Hanoi/. – Date of access: 20.01.2020.
4. Crystal benchmark [Electronic resource] – Mode of access: https://github.com/olegmikhnovich/cr-rb-c-benchmark/. – Date of access: 20.01.2020.