Treball dirigit LP

Crystal

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Hash: 36367
Quadrimestre de primavera 2022

Índex

1. Introducció

En el següent document s'explicarà les característiques i propietats més rellevants del llenguatge de programació *Crystal*. Es farà una anàlisis tant del propòsit d'aquest llenguatge, les característiques que el diferencien dels altres llenguatges, com del seu origen i història. Addicionalment, el document estarà acompanyat d'exemples il·lustratius del codi del llenguatge per a poder entendre-ho millor i inclou també una petita visió personal i crítica sobre aquest.
Finalement, aquest document es complementa amb una presentació audiovisual.

2. Descripció i principals aplicacions

Crystal és un llenguatge de programació de codi obert, això vol dir que qualsevol persona es pot descarregar i utilitzar el llenguatge de programació Crystal de forma gratuïta des de la seva pàgina web oficial.
Aquest llenguatge té una llicència Apache 2.0 i es pot utilitzar en diferents sistemes operatius com ara Linux, macOS, FreeBSD i OpenBSD. Els programadors poden executar el llenguatge de programació Crystal a les plataformes IA-32 (i386), AArch64 i X86-64.
Com Crystal és un llenguatge de programació multipropòsit o de propòsit general (s'explicarà més endavant), significa que pot ser utilitzat per desenvolupar diferents tipus d'aplicacions i programes.
Hi ha diverses empreses, com WeTransfer, Stack Goals, Labs, GigSmart, s21g, Diploid, etc., que utilitzen el llenguatge de programació Crystal per a les seves necessitats empresarials.
Una aplicació de Crystal és la **programació de baix nivell i enllaços C**, ja que Crystal té moltes funcions que el fan útil per a aplicacions incrustades i IoT (Internet of Things).
Crystal també pot funcionar amb un nombre creixent de bases de dades SQL i NoSQL. Entre ells hi ha SQLite, MySQL (o MariaDB), Postgres, MongoDB, Redis, i ReThinkDB. Els desarrolladors de Crystal es van adonar de la importància de l'accés a la base de dades per a un llenguatge de programació, de manera que van crear un mòdul de base de dades en un paquet anomenat **crystal-db**, que proporciona una API de base de dades unificada comuna. Crystal-db funciona perfectament juntament amb el controlador SQLite, MySQL i Postgres, no cal exigir-ho explícitament, però per a altres bases de dades, s'ha d'afegir controladors específics.
Finalment, aquest llenguatge també permet **la creació d'aplicacions web**.

3. Història de Crystal i relació amb LPs similars

El llenguatge de Crystal va ser anomenat originalment Joy i es va crear el **2011**, amb els objectius originals de combinar la productivitat de Ruby amb la velocitat i seguretat d'un llenguatge de tipus compilat com C. Per tant, els dos objectius fonamentals que van tenir en compte durant la creació de Crystal van ser:

- “Combinar l’eficiència de Ruby per escriure codi i l’eficiència de C per executar codi”.
- “Que el compilador comprengui el que volem dir sense haver d’especificar el tipus per a tot arreu”

També va ser influenciat, però en menor part, pels llenguatges de programació Python, Rust i Go. La primera versió oficial es va llançar el 2014 i l’última és la versió 1.4.1, llençat el 22 d’abril de 2022. Va ser dissenyat pels informàtics Ary Borenszweig, Juan Wajnerman i Brian Cardiff i desenvolupat per l’empresa Manas Technology Solutions.

Degut això, la sintaxis de Crystal és molt semblant a la de Ruby. Però encara que s'assembla al llenguatge Ruby en la seva sintaxi, Crystal compila un codi natiu molt més eficient usant un backend **LLVM**, a costa d'excloure els aspectes dinàmics de Ruby.

4. Propòsit del llenguatge

Crystal és un llenguatge de programació de **propòsit general**, és a dir, un llenguatge informàtic que s’aplica àmpliament a tots els dominis d’aplicació i no té característiques especialitzades per a un domini determinat, i els procediments, instruccions i estructures de dades dels quals estan dissenyats per a resoldre qualsevol tipus de problema.

5. Paradigmes de programació que admet Crystal

El llenguatge de programació Crystal és un llenguatge de programació de **paradigma múltiple** que admet diferents paradigmes com ara el **paradigma concurrent** i **l’orientat a objectes (Object Oriented Programming, OOP)**.

Crystal és un llenguatge de programació orientat a objectes pur, és a dir, un model de programació informàtica que organitza el disseny de software al voltant de dades i objectes, en comptes de funcions i lògica. I segueix tots els conceptes de l'enfocament de programació orientada a objectes com la classe, l'herència, l'abstracció, el polimorfisme i l'encapsulació, etc.

Que sigui un llenguatge de programació concurrent vol dir que pot executar el programa segons un ordre parcial o desordre, però sense afectar el resultat final i utilitza tècniques de paral·lelisme entre tasques com també de sincronització i comunicació. L'avantatge de la programació concurrent és el fet de poder compartir recursos entre els diferents subsistemes i optimitzar l'ús de recursos. Però s'ha d'anar en compte amb l'ordre d'execució per tal de no alterar el resultat final.

6. Sistema d'execució

Crystal és un **llenguatge de programació compilat**, la qual cosa significa es diferencien les etapes de compilació i execució, i per tant, necessita un compilador per compilar i convertir els codis en llenguatge llegible per màquina.
El codi (programa font) mitjançant un compilador es transforma en codi objecte si no existeix cap error de sintaxis. I després es monta en un executable. Sol ser eficient i es distribueix l'executable.

Figura 1: Flux de compilació i execució dels llenguatges de programació compilat

7. Sistema de tipus

Un sistema de tipus és un conjunt de regles que assignen tipus als elements d'un programa (com ara variables, expressions, funcions, etc.) per evitar errors.
La comprovació de tipus verifica que les diferents parts d'un programa es comuniquin adequadament en funció dels seus tipus.

En referència al tipatge del llenguatge crystal, és un LP de **tipat fort** (strong typing), ja que imposa restriccions que eviten barrejar valors de diferents tipus (conversions explícites).

Quan s'intenta fer la suma d'un enter amb un char, dóna el següent error de tipatge:

```
error in line 1
Error: no overload matches 'Int32#+' with type Char
```

Figura 2: Error de tipatge

La comprovació de tipus de crystal és **mixt**, ja que els tipus de variables es poden conèixer en temps de compilació o d'execució. Si s'especifica el tipus de la variable, llavors, es coneix en temps de compilació, però, si no s'especifica el tipus, la **inferència de tipus** ja ho fa pel programador i es coneixerà durant el temps d'execució.

A més és un llenguatge de **tipat nominal**, cosa que vol dir que dos tipus són compatibles només si tenen el mateix nom.

La filosofia de Crystal és requerir tan poques restriccions de tipus com sigui possible. No obstant això, es requereixen algunes restriccions.

Per exemple, es considera la següent definició de classe:

```
class Person
  def initialize(@name)
    @age = 0
  end
end
```

Figura 3: Definició d'una classe Person

Podem veure ràpidament que *@age* és un enter, però no podem saber el tipus de *@name*. El compilador pot inferir el tipus de tots els usos de la classe *Person*, però, fer-ho pot ocasionar alguns problemes:

- El tipus no és obvi per a una persona que llegeix el codi: també haurien de verificar tots els usos de *Persona* per a descobrir-ho.
- Algunes optimitzacions del compilador, com per exemple, analitzar un mètode una sola vegada, o la compilació incremental, són gairebé impossibles de fer.

A mesura que creix la complexitat del codi, aquestes qüestions poden adquirir encara més rellevància: la comprensió d'un projecte es fa més difícil, i els temps de compilació esdevenen insuportables.

Per aquesta raó, Crystal necessita saber, d'una manera òbvia, els tipus d'instància i les variables de classe. Per tal de conseguir-ho, hi ha diverses maneres de fer saber això a Crystal:

1. Amb restriccions de tipus: la forma més fàcil, però probablement la més tediosa, és fer servir restriccions de tipus explícit, és a dir, indicant explícitament el tipus de cada variable.

```
class Person
  @name : String
  @age : Int32
  def initialize(@name)
    @age = 0
  end
end
```

Figura 4: Definició d'una classe Person amb tipus

2. Sense restriccions de tipus: asignar un valor literal, és a dir, inicialitzar les variables amb valors inicials que siguin del tipus requerit.

```
class Person
  def initialize()
    @name = "John"
    @age = 0
  end
end
```

Figura 5: Definició d'una classe Person inicialitzant els valors

   En aquest cas, amb la inicialització dels valors, es pot deduir fàcilment que *@name* és un String y *@age* és un enter.

Per tant com a conclusió, tot i que el compilador pot utilitzar la inferència de tipus per a minimitzar la necessitat d'escriure tipus explícitament, es pot explicitar alguns tipus per a poder accelerar els temps de compilació i facilitar-ne la lectura i comprensió del codi.

8. Altres característiques particulars

El llenguatge crystal permet la utilització de macros, que són mètodes que reben nodes AST en temps de compilació i produeixen codi que s'enganxa en un programa. Per exemple:

```
macro define_method(name, content)
  def {{name}}
    {{content}}
  end
end

# This generates:
#
#     def foo
#       1
#     end
define_method foo, puts 1

foo # => 1

define_method foo2, puts "HOLA"

foo2 # => "HOLA"
```

Figura 6: Macros

Es pot fer servir macros per a crear diversos funcions, en l'exemple anterior s'ha creat una funció anomenada "foo" que imprimeix "1" i una altra funció "foo2" que imprimeix "HOLA" utilitzant la macro "define_method".
Les macros declarades al nivell superior són visibles a qualsevol lloc. Si una macro de nivell superior està marcada com a privada, només es pot accedir en aquest fitxer. També es poden definir en classes i mòduls, i són visibles en aquests àmbits.
Quan s'escriu codi, de vegades ens donem compte que estem escrivint mètodes gairebé duplicats els uns dels altres, que només difereixen en nom o paràmetres. En aquests casos, pot ser útil generar tot aquest codi automàticament mitjançant una versió de macro del mètode. Per tant, la utilització de macros pot ajudar a millorar la lectura i comprensió del codi.

Una altra característica important és que el nom d'una classe ha de començar per una lletra majúscula, sinó dona error de compilació.

9. Exemples de codi il·lustratius

Comanda per a compilar: crystal build nom_fitxer.cr

Comanda per a executar: ./nom_fitxer

Exemple 1: Hello World

```
class Greeter
  def initialize(@name : String )
  end

  def salute
    puts "Hello #{@name}!"
  end
end

g = Greeter.new("world")
g.salute
```

Figura 7: Exemple Hello World

Output = “Hello world!”

També es podria fer directament puts “Hello world!”, que donaria el mateix output.

Exemple 2: càlcul del nombre de fibonacci

```
class Fibonacci
  def calculo_valor(n)
    if (n == 0)
      return 0
    elsif (n == 1)
      return 1
    else
      return calculo_valor(n - 1) + calculo_valor(n - 2)
    end
  end
end

f = Fibonacci.new()
puts f.calculo_valor(2)  #1
puts f.calculo_valor(3)  #2
puts f.calculo_valor(4)  #3
puts f.calculo_valor(5)  #5
puts f.calculo_valor(6)  #8
```

Figura 8: Exemple fibonacci

Exemple 3: Valor absolut

```
class AbsValue
  def calculo_valor(n)
    if (n > 0)
      return n
    else
      return n * -1
    end
  end
end

f = AbsValue.new()
puts f.calculo_valor(2)      #2
puts f.calculo_valor(-2)     #2
```

Figura 9: Exemple valor absolut

10. Visió personal i crítica de Crystal

Crystal va ser un intent de combinar el llenguatge de programació Ruby amb els llenguatges C/C++. Al ser un llenguatge compilat estàticament, és un llenguatge ràpid, que pot assolir velocitats properes a les de C i a més, a diferència de C o C++, té la possibilitat d'utilitzar macros d'una manera molt més eficient.

Language	Version	Result(s)
C	gcc 4.9.2	3.575
Crystal	v0.20.1	6.981
Rust	1.14.0	7.808
Julia	v0.5.0	9.657
LuaJIT	2.1.0-beta2	10.150
Go	go1.7.4	10.249
Node.js	v7.3	15.122

Figura 10: velocitat d'executar Fibonacci amb diferents llenguatges de programació

Personalment, crec que és un llenguatge bastant fàcil d'aprendre i d'entendre, ja que té moltes similituds amb altres llenguatges més consolidats com C o C++ i la seva sintaxis és fàcil de llegir i simple. Finalment crec que està adquirint bastant popularitat recentment i té un gran potencial per a consolidar-se com a uns dels llenguatges de programació populars.

11. Descripció i qualitat de les fonts d'informació

La majoria de les fonts emprades han sigut extretes d'Internet i del llibre *Programming Crystal: Create High-Performance, Safe, Concurrent Apps*. La majoria del contingut relacionat amb la descripció i història del llenguatge s'han extret de diferents articles d'Internet, incloent la *Wikipedia* i la pàgina oficial de Crystal. El propòsit, els paradigmes, el sistema d'execució i el sistema de tipus han sigut recopilades mitjançant articles d'Internet, les transparències de teoría de LP de Jordi Petit i del llibre *Programming Crystal: Create High-Performance, Safe, Concurrent Apps* dels autors Ivo Balbaert i Simon St. Laurent.
Finalment els exemples de codi han sigut fetes personalment per jo mateix.
El fet d'haver consultat diverses fonts d'informació (llibre, articles d'Internet, Wikipedia…), considero que la informació trobada té un alt grau de certesa.

12. Bibliografia

Dimnet, Thomas (Gener de 2022) <A Tale of Two Engineers Discovering the Crystal Programming Language>: https://betterprogramming.pub/a-tale-of-two-engineers-discovering-the-crystal-programming-language-104b1fdbe525 (consulta: 25 d'abril de 2022)

Pàgina web oficial de Crystal (última actualització: 22 d'abril de 2022) <CRYSTAL>: https://crystal-lang.org/ (consulta: 1 de maig de 2022)

Percy, Robin (23 d'agost de 2017) <The Highs & Lows of Crystal>: https://auth0.com/blog/an-introduction-to-crystal-lang/ (Consulta: 27 d'abril de 2022)

Wikipedia <Crystal(programming language>: https://en.wikipedia.org/wiki/Crystal_(programming_language) (Consulta: 24 de maig de 2022)

Anònim (6 de juliol de 2021) <Crystal programming language: History, Features and Applications>: https://www.answersjet.com/2021/07/crystal-programming-language-history-features-applications-why-should-learn-crystal-lang.html (consulta: 1 de maig de 2022)

Balbaert, Ivo; St. Laurent, Simon. (19 de febrer 2019). *Programming Crystal: Create High-Performance, Safe, Concurrent Apps*. Andrea Stewart.
PART I. Getting Started: 1. Diving into Crystal.
PART I. Getting Started: 2. Crystal foundations.