My 7 programming ur-language

[0] The seven programming ur-languages mesupakan sebuah sebuah tulisan web yang ditulis oleh Frederick J. Ross mengenai, dalam pemrograman kita hanya perlu mengetahui 7 tipe bahasa saja, dalam perkembangan teknologi dini hari banyak sekali bahasa program yang baru, terkadang beberapa orang sulit untuk mempelajarinya namun ada beberapa orang yang mudah memahaminya juga, dan menurut saya mempelajari bahasa baru itu mudah karna biasanya bahasa baru menyalin cara bahasa yang lama karna itu jika anda ingin mempelajari bahasa pemrograman saya sarankan hanya mempelajari 7 tipe bahasa ini.

Too Long did't read

berikut listnya

1. ALGOL: C99
2. LISP: Common Lisp (SBCL)
3. ML: Haskell (GHC)
4. Self: Pharo
5. Forth: UXN
6. APL: BQN
7. PROLOG: Strands (FLeng)

ALGOL

di dalam tulisan web Frederick J. Ross tipe bahasa ini berasal dari keluarga [1] ALGOL, karakteristik pada bahasa ini adalah penggunaan loop, merubah variabel dan kondisional, bahasa ini juga merupakan bahasa yang simpel karena penulisan program sangat mirip dengan cara kerja mesin, yaitu mengeksekusi baris perintah satu persatu, karena cara kerja yang seperti mesin tipe bahasa ini juga merupakan yang paling tua dibanding yang lain, contoh bahasa dalam keluarga ini adalah, Assembly, C, Fortran, Pascal, GO, Hare, dll

dalam keluarga ini bahasa pilihan saya adalah C, karna di C saya bisa bebas melakukan apa saja, termasuk menembakan peluru ke kaki saya, berikut adalah contoh menghitung Faktorial di bahasa C

```c
#include <stdio.h>

unsigned long long
fac(unsigned int N)
{
        int acc = 1;
        for(int i = 1; i <= N; i++)
                acc *= i;
        return acc;
}

int
main(void)
{
        unsigned int N = 5;
        unsigned long long f = fac(N);
        printf("%u! = %llu\n", N, f);
        return 0;
}
```

LISP

bahasa berikutnya adalah lisp, lisp bahasa yang di desain oleh John McCarthy dengan papper berjudul [2] "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I" namun orang pertama kali yang mengimplementasikannya adalah Steve Rusell untuk IBM 704, Lisp merupakan singkatan dari List Procesing, desin pertama lisp menggunakan [3] M-expression namun kemudian menggunakan [4] S-expression, bahasa Lisp sangatlah terkenal sampai sebuah hardware dibuat hanya untuk bahasa ini, namun karena ada [5] AI Winter, mesin ini tidak lagi diproduksi, saat ini bahasa Lisp terpisah menjadi dua, yaitu [6] Scheme dan Common Lisp, Scheme lebih mengarah ke bahasa fungsional dan Common Lisp lebih mengarah ke paradigma general, salah satu implementasi dari Scheme adalah Guile, MIT Scheme, Racket, Chicken, dll, sementara untuk Common Lisp ada SBCL, ABCL, ECL, Clasp, dll.

dalam keluarga ini bahasa pilihan saya adalah Common Lisp Implementasi dari [7] SBCL, karena menurut saya SBCL Common Lisp memiliki kecepatan yang bagus, Memperogram menggunakan Common Lisp juga sangat menyenangkan karena kita dapat mengevaluasi kapan saja, jika ada kesalahan kita akan ditampilkan tampilan debugger yang interaktif, SBCL juga langsung mengkompilasi kode ke bahasa mesin, jadi sangat cepat dan saya juga memiliki pengalaman Game Development menggunakan Trial Game Enggine, dan saya sangat senang menggunakanya, untuk contoh kecil berikut kode faktorial dalam bahasa lisp, karena tidak semua implementasi Common Lisp melakukan Tail Call Optimization saya menggunakan gaya Imperatif dan Common lisp merupakan bahasa interaktif jadi kita bisa membuat fungsi dan memanggilnya langsung

```lisp
(defun factorial (n)
  (loop for i from 1 to n
        for result = 1 then (* result i)
        finally (return result)))

(format t "~a" (factorial 5))
```

ML

dari semua bahasa program, ini lah salah satu bahasa yang membuka pikiran saya, bahasa keluarga [8] ML merupakan bahasa fungsional dimana semua yang kita lakukan menggunakan fungsi, tidak hanya itu fungsi dapat memiliki argumen fungsi dan return fungsi ini biasa dinamakan currying, dalam bahasa fungsional biasanya di bedakan menjadi dua, yaitu pure dan impure, bahasa yang pure adalah bahasa yang tidak dapat mengubah suatu variable dan sebaliknya, dalam bahasa pure bahasa program yaitu Haskell, Clean, Idris, dan dalam impure ada OCaml, F#, dll, dan dalam bahasa pure kita menggunakan fungsi rekursif karena dalam pure bahasa pure fungsi tidak memiliki for loop atau while loop

bahasa pilihan saya yaitu bahasa pure [9] Haskell, bahasa yang sangat elegan namun menurut saya memiliki kekurangan yaitu package manager dalam haskell ada dua yaitu [10] Stack dan [11]Cabal, dua package manager ini memiliki sejarah panjang, package pilihan saya yaitu Cabal karna Official, seperti biasa saya memberikan contoh program factorial

```
factorial :: (Eq t, Num t) => t -> t
factorial n = go n 1

go :: (Eq t, Num t) => t -> t -> t
go 1 a = a
go n a = go ( n - 1 ) ( a * n )
```

kode diatas merupakan versi tail recursion yang lebih efisien dari yang biasa

Self

self sama dengan [12] object oriented programming, saat ini bahasa object oriented programming yang paling popular adalah Java namun saya rasa java bukan lah pure object oriented, yang saya maksud object oriented itu sama dalam yang ada dalam bahasa [13] SmallTalk, dan SmallTalk merupakan bahasa pertama yang menggunakan konsep object oriented programming, untuk saat ini yang saya tau hanya ada 2 bahasa program yang masih menggunakan paradigma pure object oriented yaitu [14] Squeak dan [15] Pharo, dari kedua itu saya mencoba Pharo, ketika saya menjalankan Pharo saya seperti berada dalam VM Qemu, setelah mencoba beberapa minggu programming di Pharo sangat bagus karna tools yang disediakan berupa gui, jadi kita bisa bisa melihat/menginspeksi semua class dengan gui.

dalam bahasa pure OOP, ada konsep yang dinamakan message passing, mirip dengan aktor model, jadi untuk berinteraksi dengan klass kita mengirim sebuah pesan ke klass tersebut, seperti ini contoh jika kita ingin menghubungi Objek Transcrip dan menggunakan method show, dimana digunakan untuk printing sebuah pesan ke jendela Transcript

```
Transcript show: 'Hello Pharo!'.
```

saya sudah lama tidak menggunakan Pharo di lain waktu akan saya update halaman ini

Forth

hampir mirip dengan LISP [16] forth bekerja pada sebuah stack, forth pertama kali digunakan untuk embeded machine karna forth merupakan bahasa yang simpel dan mini, di keluarga ini saya pernah menulis kode forth dengan implementasi dari gnu, namanya adalah [17] gnuforth, gnuforth ini sangat portabel sampai bisa di jalankan di ponsel android, namun karna lisensi yang kurang saya suka maka saya mencari alternative, bertemulah UXN, [18] UXN merupakan sebuah bahasa program untuk mesin Varvara, Varvara ini di targetkan untuk mesin mini, namun memiliki fungsionalitas gui, saya juga pernah menjalankannya di Nintendo 3DS saya, dan berjalan dengan lancar, karna portabel dan simpel inilah saya memilih UXN.

seperti inilah wujud hello world dari UXN

```
|10 @Console &vector $2 &read $1 &pad $5 &write $1 &error $1

|100

@on-reset ( -> )
        ;my-string print-text
        BRK

@print-text ( str* -- )
        &while
                LDAk .Console/write DEO
                INC2 LDAk ?&while
        POP2
        JMP2r

@my-string
        "Hello 20 "World! 00
```

APL

A Programming Language, [19] APL sama dengan Forth dan Lisp, tetapi APL bekerja pada Array, di dalam [20] Array Programming jika kita menambah angka dua dalam sebuah Array maka akan mempengaruhi seluruh isi dari Array tersebut, yang paling terkenal dari keluarga APL adalah penggunaan symbol yang mungkin memiliki berbeda arti dalam konteks tertentu, dalam bahasa ini saya pernah mencoba bahasa Dyalog APL dan BQN, Dyalog APL merupakan bahasa Non-Free sebaliknya dengan BQN jadi saya memilih BQN disini, sebelumnya untuk keluarga ini saya tidak tertarik dikarenakan penggunaan symbol namun karna saya menonton youtube dari code_report saya rasa keluarga dari APL itu keren jika kita tau apa maksud dari simbol yang digunakan, untuk merasakan kode BQN berikut kodennya, diambil dari [21] github codereport Contest 336

```
MaxScore <- +´0<·+‘∨

# Tests
MaxScore (2,-1,0,1,-3,3,-3) # 6
MaxScore (-2,-3,0)          # 0
```

dan ini untuk bahasa Haskell

```
import Data.List (sort)

maxScore = length
         . filter (>0)
         . scanr1 (+)
         . sort
```

Prolog

prolog merupakan singkatan dari [22] Programming Logic, dimana kita menggunakan logic untuk menentukan semuanya, salah satu fitur yang saya suka dari prolog adalah Backtracking, jadi ada sebuah rule dapat menghasilkan banyak output kita bisa memilihnya di repl, di dalam prolog tidak ada yang namanya fungsi, yang ada hanyalah fact dan rule, di prolog juga menggunakan koma dan semikolon untuk percabangan, sama dengan Lisp prolog merupakan bahasa yang di kembangkan untuk AI, namun saya rasa Prolog berhasil selamat dari AI Winter, Prolog juga merupakan bahasa standar seperti Lisp, salah satu implementasi yang saya suka adalah [23] trella prolog, namun pada keluarga ini ada jenis bahasa yang unik yaitu [24] Concurent Prolog System, contohnya adalah Strands Prolog, dimana kita bisa menjalankan perintah secara konkuren dan cepat, namun pada concurent prolog sepertinya sudah banyak ditinggalkan dan saya tidak tau kenapa akan saya update jika mengetahuinya, salah satu implementasi compiler yaitu [25] Fleng yang dikembangkan oleh pembuat [26] Chicken Scheme yaitu Felix Winkelmann, bahasa fleng ini tidak memiliki interpreter melakukan kompilasi langsung ke bahasa mesin, berikun adalah contoh bahasa Strands

```
-initialization(main).
-uses([unix, fmt]).

main :-
  unix:unix([
    connect(inet('127.0.0.1', 8080), C)
  ]),
  handle(C).

handle(normal(C)) :-
  C = [
    fread(256, R)
  ],
  fmt:format('~s', [R]).
```

Referensi

[0] *The seven programming ur-languages* https://madhadron.com/programming/seven_ur_languages.html

[1] *ALGOL* https://en.wikipedia.org/wiki/ALGOL

[2] *Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I* https://www-formal.stanford.edu/jmc/recursive.pdf

[3] *M-expression* https://en.wikipedia.org/wiki/M-expression

[4] *S-expression* https://en.wikipedia.org/wiki/S-expression

[5] *AI Winter* https://en.wikipedia.org/wiki/AI_winter

[6] *Scheme* https://en.wikipedia.org/wiki/Scheme_(programming_language)

[7] *SBCL: Steel Bank Common Lisp* http://www.sbcl.org

[8] *ML* https://en.wikipedia.org/wiki/ML_(programming_language)

[9] *Haskell Language* https://www.haskell.org

[10] *The Haskell Tool Stack* https://docs.haskellstack.org/en/stable/

[11] *Cabal: Common Architecture for Building Applications and Libraries* https://www.haskell.org/cabal/

[12] *Object Oriented Programming* https://en.wikipedia.org/wiki/Object-oriented_programming

[13] *Smalltalk* https://en.wikipedia.org/wiki/Smalltalk

[14] *Squeak/Smalltalk* https://squeak.org

[15] *Pharo: The immersive programming experience* https://pharo.org

[16] *Forth* https://forth-standard.org

[17] *GNU Forth* https://gforth.org

[18] *UXN* https://100r.co/site/uxn.html

[19] *APL* https://aplwiki.com

[20] *Array Programming* https://en.wikipedia.org/wiki/Array_programming

[21] *codereport/LeetCode* https://github.com/codereport/LeetCode

[22] *Logic Programming* https://en.wikipedia.org/wiki/Logic_programming

[23] *trella prolog* https://github.com/trealla-prolog/trealla

[24] *Concurrent logic programming* https://en.wikipedia.org/wiki/Concurrent_logic_programming

[25] *Fleng* https://www.call-with-current-continuation.org/fleng/fleng.html

[26] *Chicken Scheme* https://www.call-cc.org