**RYBE**

A suite of technologies that empower software engineers the capacity to engineer softwares utopianly.

THE OFFICIAL REFERENCE

CONCEPT 1

INFORMATION CREATION TECHNIQUE

GROTESQUE TECHNIQUE

Elements

========

Alpha and Beta

Sequence

========

ABABABBA

CONVENIENCE TECHNIQUES

CARDINAL TECHNIQUE

Elements

========

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Sequence

========

[c: 254145748544473964343847038343]

[c: 548754890578475498578475583434]

UNICODE TECHNIQUE

Elements

========

Elements of Unicode

Sequence

========

[u: Hello world!]

[u: Hello world! This is Ryb-9! []]

CONCEPT 2

MEMORY MANAGEMENT IN COMPUTER

gtmc || Get total memory capacity

==== The concept of yield

@\*-%-[sequence] || Yield set

==== (1: Total memory capacity)

==== Secondary relation (-) and Attribute (!)

==== 1st yield (A), 2nd yield (B), 3rd yield (AA),

==== ==== +InfoXYZ![-1/+1], (+InfoXYZ![-1/+1])

==== 1st yield: yield collection’s cardinality

==== ==== Language created information (LCI)

==== ==== Programmer created information (PCI)

==== 2nd yield: execution completion status type

==== ==== A: Execution suspended: Error encountered: Language error

==== ==== B: Execution suspended: Suspended by imported program

==== ==== ==== Added for Concept X

==== ==== AA: Execution completed

==== 3rd yield: execution completion status (description)

nlin @InstXYZ || Name last instruction

@InstXYZ-%-[sequence]![-1/+1] || Yield set

gamc || Get available memory capacity

==== 1: Available memory capacity

nlin @InstXYZ || Name last instruction

@InstXYZ-%-[sequence]

CONCEPT 3

COMPUTER FUNDAMENTAL CAPABILITIES

SECTION 3.1

INFORMATION CREATION IN COMPUTER

Maximum of 92 characters per line.

crte +InfoXYZ

==== Name: “+” + Uppercase English alphabet + alpha-numeric

==== ==== Max 48 characters

+InfoXYZ: asqn [position (:position)..., sequence (:sequence)...]

==== The concept of seed.

==== position: -1 (AAA), +1 (ABA), ! (BAA), +InfoXYZ![-1/+1],

==== ==== (+InfoXYZ![-1/+1])

==== sequence: AAAB, +InfoXYZ![-1/+1], (+InfoXYZ![-1/+1])

==== ==== ==== ADDITIONAL ==== ==== ====

+InfoXYZ: gcrd || Get cardinality

nlin @InstXYZ || Name last instruction

@InstXYZ-%-[sequence]

SECTION 3.2

INFORMATION CHANGING IN COMPUTER

+InfoXYZ: rsqn [target sequence- first element- position, no of elements]

==== target sequence- first element- position: 1st (A), 2nd (B), 3rd (AA) ==== ==== +InfoXYZ![-1/+1], (+InfoXYZ![-1/+1])

==== no of elements: 1 (A), 2 (B), 3 (AA), +InfoXYZ![-1/+1],

==== ==== (+InfoXYZ![-1/+1])

SECTION 3.3

INFORMATION EXCHANGE IN COMPUTER

ecib || Check incoming sequence buffer occupation status

==== 1: Incoming buffer occupation status

nlin @InstXYZ

@InstXYZ-%-[A]

eiis || Import incoming sequence

==== 1: Message Sequence

nlin @InstXYZ

@InstXYZ-%-[A]

ecob || Check outgoing sequence buffer occupation status

==== 1: Outgoing buffer occupation status

nlin @InstXYZ

@InstXYZ-%-[A]

eeos [sequence] || Export outgoing sequence

SECTION 3.4

INFORMATION PRESERVATION IN COMPUTER

pcib || For information preservation, check incoming sequence

|| buffer occupation status

==== 1: Incoming buffer occupation status

nlin @InstXYZ

@InstXYZ-%-[A]

piis || For information preservation, import incoming sequence

==== 1: Message Sequence

nlin @InstXYZ

@InstXYZ-%-[A]

pcob || For information preservation, check outgoing sequence

|| buffer occupation status

==== 1: Outgoing buffer occupation status

nlin @InstXYZ

@InstXYZ-%-[A]

peos [sequence] || For information preservation, export outgoing

|| sequence

SECTION 3.5

HUMAN-COMPUTER COMMUNICATION

ccib || For human-computer communication, check incoming sequence

|| buffer occupation status

==== 1: Incoming buffer occupation status

nlin @InstXYZ

@InstXYZ-%-[A]

ciis || For human-computer communication, import incoming sequence

==== 1: Message Sequence

nlin @InstXYZ

@InstXYZ-%-[A]

ccob || For human-computer communication, check outgoing sequence

|| buffer occupation status

==== 1: Outgoing buffer occupation status

nlin @InstXYZ

@InstXYZ-%-[A]

ceos [sequence] || For human-computer communication, export outgoing

|| sequence

CONCEPT 4

COMPLETE INSTRUCTION CREATION (& REPETIION)

CONDITIONAL (0/1)

ecnd (+InfoXYZ![-1/+1] == +InfoXYZ![-1/+1])| instruction x

==== Unless data is typed

==== &, |, and ()

ecnd (+InfoXYZ![-1/+1] != +InfoXYZ![-1/+1])| @

!2^^

instruction w

instruction x

instruction y

instruction z

!2\*\*

==== Divisioning (Creating divisions)

==== ==== 2nd Degree Division

==== ==== Introductory boundary

==== ==== Terminal boundary

FATED (1)

eftd| instruction x

SELECTIONAL (1/1)

eslc| @

!2^^

eopt (+InfoXYZ![-1/+1] != +InfoXYZ![-1/+1])| @

!3^^

instruction w

instruction x

instruction y

instruction z

!3\*\*

eopt (+InfoXYZ![-1/+1] != +InfoXYZ![-1/+1])| @

!3^^

instruction w

instruction x

instruction y

instruction z

!3\*\*

elrs| @

!3^^

instruction w

instruction x

instruction y

instruction z

!3\*\*

!2\*\*

REPETITION

eftd| @

!2^^

instruction w

instruction x

eftd| rpat

instruction y

instruction z

!2\*\*

cest || Check existential status

==== 1: Existential status

wipe [+InfoXYZ, +InfoXY2]

wipe arin || All reptition’s information

==== Instruction not usable for premier (0th repetition)

Continuation on next line: leave 8 spaces, then start at 9th space

CONCEPT 5

INSTRUCTION-SET IMPORTATION

instruction w

$-[sequence]![-1/+1] || Seed set

==== sequence: 1st (A), 2nd (B), +InfoXYZ![-1/+1], (+InfoXYZ![-1/+1])

==== 1st seed: Seed set’s cardinality

instruction x

eftd| yeld [+InfoXYZ, +InfoXY2]

instruction y

instruction z

eftd| yeld [+InfoXY3, +InfoXY4]

eftd| sspn

eftd| +InfoXYZ: eais || Execute as an instruction set

eftd| +InfoXYZ: eais [+InfoXYZ, +InfoXY2]

eftd| nlin @InstXYZ

@InstXYZ-%-[-1/+1]

Concept 11

Program Fragment Reapplicability

Extension

Inter-level communication

Concept 13

Concurrency

Concept 10

Step Packaging

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**####-####-####-####-####-####-####-####-####-###**

**Comment**

+InfoXYZ: hrdn || Harden

+InfoXYZ: sftn || Soften

**Unit 1: Thread**

Thread:: Standard and Creation;

---- [and Premier-standard]

Program: Standard and Fragment; One-time and Repeatable;

---- Local and Imported

---- [and Premier-standard]

eofp +InfoXYZ (--seed--)

---- || Execute as a one-time imported fragment program

mrfp +InfoXYZ ($Xyz)

---- || Make a repeatable imported fragment program

mtpc +InfoXYZ (@Xyz)

---- || Make an information type repeatable imported fragment program collection

+Information

$Program

@InformationType

!LastInstruction

Unit 2: Thread Structure

^^^^ 1: RFPS ^^^^

^^^^ 2: RPFP ^^^^

N: @Hello

\*\*\*\* 2: RPFP \*\*\*\*

Xxxx

---- 2: RPFP ----

^^^^ 2: ITYP ^^^^

N: @Gello

\*\*\*\* 2: ITYP \*\*\*\*

^^^^ 3: RPFP ^^^^

T: T

N: $Hello

\*\*\*\* 3: RPFP \*\*\*\*

Xxxx

---- 3: RPFP ----

^^^^ 3: RPFP ^^^^

T: I

N: $Hello

\*\*\*\* 3: RPFP \*\*\*\*

Xxxx

---- 3: RPFP ----

---- 2: ITYP ----

---- 1: RFPS ----

^^^^ 1: LOSS ^^^^

^^^^ 2: LOSP ^^^^

N: @Main1XYZ

S: xxxx

E: xxxx

\*\*\*\* 2: LOSP \*\*\*\*

^^^^ 3: RPFP ^^^^

xxxx

xxxx

---- 3: RPFP ----

^^^^ 3: APRG ^^^^

1^^1

2^^2

2--2

xxxx

xxxx

2--2

2^^2

xxxx

xxxx

2--2

1--1

1==1

---- 3: APRG ----

---- 2: LOSP ----

I: xxxx

==== 2: LOSP ====

^^^^ 2: LOSP ^^^^

N: @Main2XYZ

\*\*\*\* 2: LOSP \*\*\*\*

xxxx

xxxx

xxxx

xxxx

---- 2: LOSP ----

---- 1: LOSS ----

**Unit 2: Premier Program**

Access to connected components

nlin !Inst ||Name last instruction

!Inst [1]

end!

**Unit 2: Instruction Type: Conditional, Fated, Selectional, and Repetitional**

Name: alphabet + alpha numeric and “ ’ ”; max 48 characters

**Unit 4: Repeatability: Repeatable Fragment Program and Information Type**

+InfoXYZ: type

+@: wpot

$SayHello: exct (--seed--)

@InformationType:$CreateAnInformation1: exct ()

+InfoXYZ:$ProcessInformation1: exct ()

+InfoXYZ:$ProcessInformation2: exct

**Unit 5: Imported Program (One-time & Repeatable (Program & Type))**

**Unit 6: Step Packaging**

A^^A

T: This is an algorithm step to be carried out.

D: Description.

I: DrawInData1, DrawInData2

A\*\*A

instruction w

instruction x

B^^B

T: This is an algorithm step to be carried out.

D: Description.

I: DrawInData1, DrawInData2

B\*\*B

instruction w

instruction x

C^^C

T: This is an algorithm step to be carried out.

D: Description.

I: DrawInData1, DrawInData2

C\*\*C

instruction w

instruction x

instruction y

instruction z

C--C

PullOutData1, PullOutData2

C==C

instruction y

instruction z

B--B

PullOutData1, PullOutData2

B==B

instruction y

instruction z

B^^B

T: This is an algorithm step to be carried out.

D: Description.

I: DrawInData1, DrawInData2

B\*\*B

instruction w

instruction x

instruction y

instruction z

B--B

PullOutData1, PullOutData2

B==B

A--A

PullOutData1, PullOutData2

A==A

instruction x

instruction y

instruction z

|| aspect

**Unit 7: Thread**

|  |  |
| --- | --- |
|  | |
| some instruction  thread {meta info (time share)} {  instruction  }  time share changeable  Execute for X time  Assign memory; assign more memory later  Get available memory quantity  || ---->  || <---- |  |

**Unit X: Rest**

Data transfer from phase/levels