AlphaCode
Language Design Specification Document

1. Language Overview

A. Introduction to the Language

AlphaCode is a programming language that adopts the NATO phonetic alphabet for its syntax, aiming to enhance clarity and minimize ambiguity in code communication. This unconventional language replaces traditional programming symbols with phonetic representations of English letters, offering a distinctive and unambiguous approach to coding. AlphaCode maintains the essential elements of a Turing-complete programming language while introducing a novel linguistic perspective.

B. Discussion of Influences

AlphaCode draws inspiration from the NATO phonetic alphabet, a system designed to ensure clear and precise communication of letters and words in challenging environments. The irony of using a system meant to eliminate confusion in communication to represent programming constructs adds a silly twist to AlphaCode. While traditional programming languages may inadvertently introduce ambiguity in the interpretation of symbols, AlphaCode embraces a linguistic approach to enhance code clarity.

C. BNF Description of Syntax

To eliminate left recursion
Convert: A -> Aa1 | Aa2 | ... | Aan | B
T o:
A -> BA'
A' -> a1 A' | a2 A' | ... | ""

Context-Free Portion
==========

< Program > ::= < Program > < Statement >
| < Statement >

< Statement > ::= OSCAR < Statement' > HOTEL
| < IO-Operation > HOTEL
| < Array-Dim > HOTEL
| < FIVEEYES-Decl > HOTEL
| < Branch > HOTEL
| < Loop > HOTEL
| < Expression > HOTEL
| HOTEL

< Statement' > ::= < Ref' > < Statement''>
< Statement'' > ::= LIMA < Expression >
| < Factor' > < Term' > < Expression' >

< Branch > ::= PAPA < Condition > HOTEL < Program > QUEBEC PAPA

< Loop > ::= ROMEO < Condition > HOTEL < Program > QUEBEC ROMEO

< Condition > ::= < Expression > < Condition' >

< Condition' > ::= SIERRA < Expression >
| UNIFORM < Expression >
| VICTOR < Expression >
| TANGO < Expression >
| LIMA < Expression >
| WHISKEY < Expression >

< IO-Operation > ::= MIKE < Expression >
| NOVEMBER OSCAR

< Array-Dim > ::= ILLUMINATI OSCAR < Expression >

< FIVEEYES-Decl > ::= FIVEEYES OSCAR < FIVEEYES-Fields > QUEBEC FIVEEYES

```
< FIVEEYES-Fields > ::= < FIVEEYES-Fields > < NATO >
                     | < NATO >

< NATO >         ::= NATO < NATO' >

< NATO' >        ::= OSCAR
                     | < Array-Dim >

< Expression > ::= < Term > < Expression' >

< Expression' > ::= ALPHA < Term > < Expression' >
                  | BRAVO < Term > < Expression' >
                  | ""

< Term >       ::= < Factor > < Term' >

< Term' >      ::= CHARLIE < Factor > < Term' >
                  | DELTA < Factor > < Term' >
                  | KILO < Factor > < Term' >
                  | ""

< Factor >     ::= < Exponent > < Factor' >

< Factor' >    ::= ECHO < Exponent > < Factor' >
                  | ""

< Exponent >   ::= < Number >
                  | BRAVO < Exponent >
                  | FOXTROT < Expression > GOLF

< Number >     ::= INDIA | JULIETT | < Ref >

< Ref >        ::= OSCAR < Ref' >

< Ref' >       ::= XRAY < Expression > YANKEE < Ref' >
                  | ZULU OSCAR < Ref' >
                  | ""
```

Lexer Grammar

Token	Actual Input Used While Coding
ALPHA	ALPHA
BRAVO	BRAVO
CHARLIE	CHARLIE
DELTA	DELTA
ECHO	ECHO
FOXTROT	FOXTROT
GOLF	Golf
HOTEL	\n
INDIA	[0-9]+
JULIETT	[0-9]+\.[0-9]+
KILO	KILO
LIMA	LIMA
MIKE	MIKE
NOVEMBER	NOVEMBER
OSCAR	[a-zA-Z_][a-zA-Z_0-9]*
PAPA	PAPA
QUEBEC	QUEBEC
ROMEO	ROMEO
SIERRA	SIERRA
TANGO	TANGO
UNIFORM	UNIFORM
VICTOR	VICTOR
WHISKEY	WHISKEY
XRAY	XRAY
YANKEE	YANKEE
ZULU	ZULU
ILLUMINATI	ILLUMINATI
FIVEEYES	FIVEEYES
NATO	NATO

NATO	Meaning in Mathematical Form
ALPHA	+
BRAVO	-
CHARLIE	*
DELTA	/

ECHO	^
FOXTROT	(
GOLF	)
HOTEL	\n
INDIA	[0-9]+
JULIETT	[0-9]+\.[0-9]+
KILO	MOD
LIMA	=
MIKE	display
NOVEMBER	input
OSCAR	[a-zA-Z_][a-zA-Z_0-9]*
PAPA	if
QUEBEC	end
ROMEO	while
SIERRA	>
TANGO	>=
UNIFORM	<
VICTOR	<=
WHISKEY	<>
XRAY	[
YANKEE	]
ZULU	.
ILLUMINATI	dimension
FIVEEYES	record
NATO	field

2. Example Programs

A. Hello World

```
MIKE “Hello, World!”
```

B. Implementation and Test of Bubble Sort

```
ILLUMINATI arr[5]
NOVEMBER arr[0]
NOVEMBER arr[1]
NOVEMBER arr[2]
NOVEMBER arr[3]
NOVEMBER arr[4]

MIKE "Original array:"
MIKE arr[0]
MIKE arr[1]
```

```
MIKE arr[2]
MIKE arr[3]
MIKE arr[4]

ILLUMINATI n
LIMA n CHARLIE 5 5

ILLUMINATI i
LIMA i 0

ILLUMINATI j
LIMA j 0

ROMEO UNIFORM i BRAVO n 1
  ROMEO UNIFORM j BRAVO n i
    PAPA SIERRA arr[j] arr[ALPHA j 1]
      ILLUMINATI temp
      LIMA temp arr[j]
      LIMA arr[j] arr[ALPHA j 1]
      LIMA arr[ALPHA j 1] temp
    QUEBEC PAPA
    LIMA j ALPHA j 1
  QUEBEC ROMEO
  LIMA i ALPHA i 1
QUEBEC ROMEO

MIKE "Sorted array:"
MIKE arr[0]
MIKE arr[1]
MIKE arr[2]
MIKE arr[3]
MIKE arr[4]
```

C. Tower of Hanoi Solver

```
ILLUMINATI towerA 3

ILLUMINATI towerB 3

ILLUMINATI towerC 3

# Initialize towers with disks

towerA XRAY 0 YANKEE LIMA 3

towerA XRAY 1 YANKEE LIMA 2
```

```
towerA XRAY 2 YANKEE LIMA 1

towerB XRAY 0 YANKEE LIMA 0
towerB XRAY 1 YANKEE LIMA 0
towerB XRAY 2 YANKEE LIMA 0

towerC XRAY 0 YANKEE LIMA 0
towerC XRAY 1 YANKEE LIMA 0
towerC XRAY 2 YANKEE LIMA 0



temp LIMA 0
i1 LIMA 2
i2 LIMA 0
i3 LIMA 0
numDisks LIMA 3
totalMoves LIMA 2 ECHO numDisks BRAVO 1

moveCount LIMA 1
ROMEO moveCount VICTOR totalMoves
 val LIMA moveCount KILO 3
 PAPA val LIMA 1
  PAPA towerA XRAY i1 YANKEE SIERRA towerC XRAY i3 YANKEE
   temp LIMA towerA XRAY i1 YANKEE
   towerC XRAY i3 YANKEE LIMA temp
   towerA XRAY i1 YANKEE LIMA 0
   i3 LIMA i3 ALPHA 1
   i1 LIMA i1 BRAVO 1
  QUEBEC PAPA
```

```
  PAPA towerA XRAY i1 YANKEE UNIFORM towerC XRAY i3 YANKEE
    temp LIMA towerC XRAY i3 YANKEE
    towerA XRAY i1 YANKEE LIMA temp
    towerC XRAY i3 YANKEE LIMA 0
    i3 LIMA i3 BRAVO 1
    i1 LIMA i1 ALPHA 1
  QUEBEC PAPA
QUEBEC PAPA

PAPA val LIMA 2
  PAPA towerA XRAY i1 YANKEE SIERRA towerB XRAY i2 YANKEE
    temp LIMA towerA XRAY i1 YANKEE
    towerB XRAY i2 YANKEE LIMA temp
    towerA XRAY i1 YANKEE LIMA 0
    i2 LIMA i2 ALPHA 1
    i1 LIMA i1 BRAVO 1
  QUEBEC PAPA

  PAPA towerA XRAY i1 YANKEE UNIFORM towerB XRAY i2 YANKEE
    temp LIMA towerB XRAY i2 YANKEE
    towerA XRAY i1 YANKEE LIMA temp
    towerB XRAY i2 YANKEE LIMA 0
    i2 LIMA i2 BRAVO 1
    i1 LIMA i1 ALPHA 1
  QUEBEC PAPA
QUEBEC PAPA

PAPA val LIMA 0
```

```
    PAPA towerB XRAY i2 YANKEE SIERRA towerC XRAY i3 YANKEE
      temp LIMA towerB XRAY i2 YANKEE
      towerC XRAY i3 YANKEE LIMA temp
      towerB XRAY i2 YANKEE LIMA 0
      i2 LIMA i2 BRAVO 1
      i3 LIMA i3 ALPHA 1
    QUEBEC PAPA

    PAPA towerB XRAY i2 YANKEE UNIFORM towerC XRAY i3 YANKEE
      temp LIMA towerC XRAY i3 YANKEE
      towerB XRAY i2 YANKEE LIMA temp
      towerC XRAY i3 YANKEE LIMA 0
      i2 LIMA i2 ALPHA 1
      i3 LIMA i3 BRAVO 1
     QUEBEC PAPA
  QUEBEC PAPA


   moveCount LIMA moveCount ALPHA 1
   MIKE "Move " moveCount ":"
QUEBEC ROMEO

MIKE "Tower A: "
MIKE towerA XRAY 0 YANKEE
MIKE towerA XRAY 1 YANKEE
MIKE towerA XRAY 2 YANKEE

MIKE "Tower B: "
MIKE towerB XRAY 0 YANKEE
```

MIKE towerB XRAY 1 YANKEE

MIKE towerB XRAY 2 YANKEE

MIKE "Tower c: "

MIKE towerC XRAY 0 YANKEE

MIKE towerC XRAY 1 YANKEE

MIKE towerC XRAY 2 YANKEE

3. Language Reference Manual

A. Operator Precedence Chart

The following table illustrates the operator precedence in AlphaCode, from highest to lowest:

Operators	**Associativity**
BRAVO (Unary minus), FOXTROT + GOLF	Right
ECHO (Exponent)	Right
KILO, CHARLIE, DELTA	Left
ALPHA, BRAVO	Left
SIERRA, UNIFORM, VICTOR, TANGO, LIMA, WHISKEY	Non-associative
LIMA (Assignment}	Left

B. Operator Explanations

- **ECHO (Exponent):** Raises the left operand to the power of the right operand.
- **FOXTROT + GOLF:** This is the left and right parenthesis, there can be an expression between the parenthesis.
- **KILO:** Computes the modulus of the left operand divided by the right operand.
- **CHARLIE, DELTA:** Multiplication and division operators.
- **ALPHA, BRAVO :** Addition and subtraction operators.
- **SIERRA, UNIFORM, VICTOR, TANGO, LIMA, WHISKEY:** Comparison operators for greater than, greater than or equal to, less than, less than or equal to, equal to, and not equal to, respectively.
- **LIMA:** For Assignment.

C. Semantic Information

Here is some information on semantics.

Arrays and Indices

- Arrays in AlphaCode are declared using the `ILLUMINATI` keyword, followed by the array name and size.
- Array indices start from 0 and can be accessed using square brackets, e.g., `arr[0]`.
- Arrays are zero-initialized upon declaration

Type Coercion Rules

- Type inference is used.
- There are only two types in this language: Integer and Real which uses the token INDIA and JULIETT respectively.
- Type conversion is done automatically for operations involving different data types depending on the operation being done. Most of the converted to Juliett (Real) if there is a type difference,

Parameter Passing Information

- At this moment there is no functions in this language, there are no parameter passing done.

Other Semantic Information

- Variables are declared variable name followed by an expression.
- Control flow statements include `PAPA` (if) and `ROMEO` (while) are declared followed by a condition statement and they using the keyword QUEBEC followed by the control flow statement used(`PAPA` (if) and `ROMEO` (while)).
- Input/output operations are performed using `MIKE` (display) and `NOVEMBER` (input).