＜論文＞

# 擬似言語解析・実行システムALEXの改訂について

# On the revision of a system called ALEX which parses and executes programs written in pseudo-language

鑰　山　　徹

**アブストラクト：**

本稿では、擬似言語プログラムを解析・実行するシステムALEXの第２版ALEX2について述べる。ALEX2では、データ型を拡張したほか、構造体を扱えるようにした。本稿では、まず、ALEXの概要を述べ、その後、改訂内容を論じる。最後に、今後の課題を論じる。

**Abstract :**

This paper describes a system called ALEX which parses and executes programs written in pseudo-language. First, the outline of the whole system is explained. Second, the revised contents of the system are described. Finally, this paper discusses some unsolved problems.

**キーワード：**

擬似言語　Java　構文解析　構造体

**Keywords :**

pseudo-language Java parsing structure

## １．はじめに

擬似言語とは、基本情報技術者試験で出題されるアルゴリズムを表現する日本語風の擬似的なプログラム言語である。擬似言語では、選択構造を表す「▲」、「▼」、反復構造を表す「■」、

代入を表す「←」などの記号を用いるほか、「変数ｘに整数値を入力する」などの日本語表現を認める。

筆者は既に、擬似言語で書かれたプログラムを解析•実行するシステムALEX（Algorithm Execution Systemの略）バージョン１（以下、ALEX1と略記する）を開発している [2] が、今回、その内容を大幅に改訂し、バージョン２を完成させた。本稿は、そのALEXバージョン２（以下、ALEX2と略記する）についての報告である。擬似言語プログラムの例を図１に示す。

```
// 「うるう年」の判定
○変数ｙｅａｒを整数型として定義する
・整数ｙｅａｒを入力する
▲ (year%4=0 かつ year%100≠0) または (year%400=0)
  ・year＋”は、うるう年”を出力する
◆                                  // 「そうでないとき」の意
  ・year　＋”は、うるう年ではない”を出力する
▼
```

図１　擬似言語プログラムの例

## ２．ALEX2の内部構造

### ２．１　サブシステム

ALEX2は解析部と実行部からなる。解析部は

- 字句解析サブシステム
- 構文解析サブシステム
- 意味解析サブシステム

からなり、実行部は

- 実行サブシステム

から構成されている（図２）。

#### 1）字句解析サブシステム

擬似言語のソースプログラムを行単位に解析し、変数名、演算子、記号などのトークンを切り出し、構文解析サブシステムに渡す。

#### 2）構文解析サブシステム

文脈自由文法を用いて、トークン列の構文の妥当性を検証する。正しい場

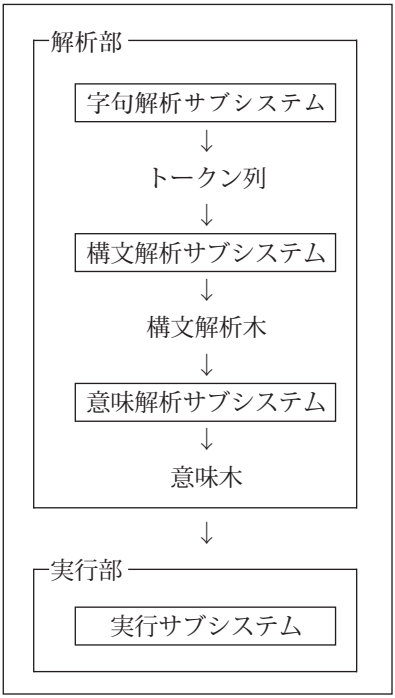


図２　サブシステムの関連

合には構文解析木を作成して、意味解析サブシステムに渡す。

**3）意味解析サブシステム**

構文解析木から、LISP形式の意味木を生成する。

意味解析までで文法エラーがなかった場合には、実行サブシステムが実行できる。

**4）実行サブシステム**

LISP形式のユーザプログラムを実行する。

図１のプログラムの場合の実行ウィンドウを図３に示す。実行ウィンドウで、連続実行ボタンをクリックすると､実行が開始される。図３において、2018と入力した場合の実行例を以下に示す。

```
・整数ｙｅａｒを入力する
---> 2018
 2018は、うるう年ではない
```

## ２．２　クラスとその機能

ALEX2は、46のクラスから構成されている。そのうちの主なものを以下に掲げる。

・ MainWindowクラス

ALEX2全体を統括するクラスで、メインウィンドウを表示する。

・ LexAnalyzerクラス

字句解析を行う。

・ LanguageProcessorクラス

構文解析を行う。

・ Semanticsクラス

意味解析を行う。

・ GrammarBaseクラス

擬似言語の文法を保持する。日本語の文法も含む。

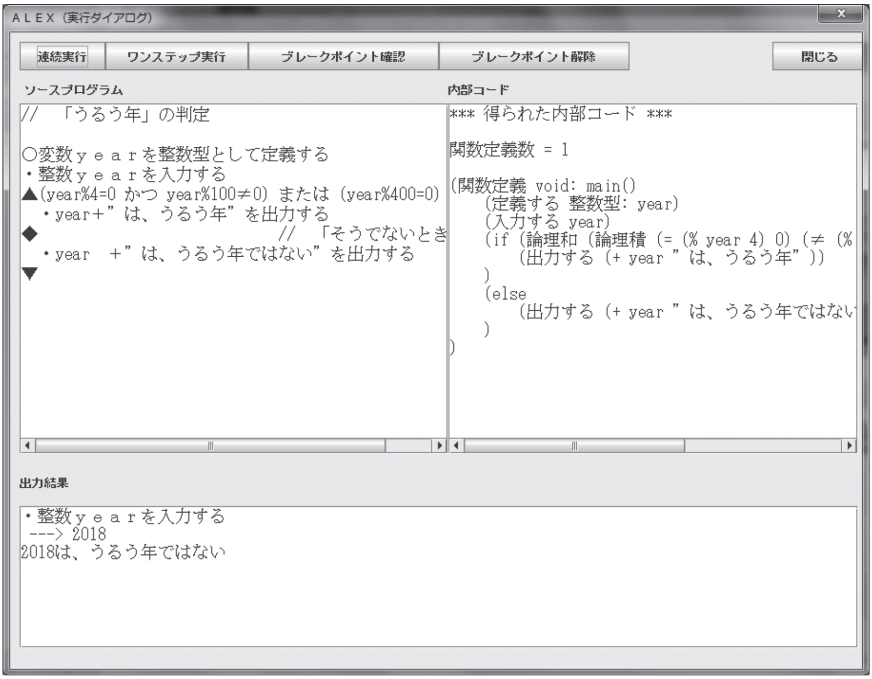


図３　実行ウインドウの例

・ DicKnowledgeBaseクラス

日本語辞書を保持し、字句解析関係の基本処理を行う。

・ Executorクラス

擬似言語プログラムの実行を行う。

## ２．３　文法構成

日本語文法を含む擬似言語の文法は、すべて「文脈自由文法」で記述されている。ALEX2は172個の文法規則を持ち、構文解析・意味解析で利用される。図４に、文法規則のいくつかを示す。図４では、ＢＮＦ記法で文法規則を表している。ＢＮＦ記法では、＜と＞で囲まれたものが文法カテゴリである。また、

左辺：：＝右辺

は左辺が右辺で定義されることを表す。例えば、

＜名詞句＞：：＝＜名詞＞＜助詞＞

は、名詞に助詞が付いたものを名詞句として定義することを意味している。

＜変数定義文＞　：：＝　○　＜データ型名＞　：　＜変数名の並び＞
＜実行文＞　：：＝　・　＜変数名＞　←　＜算術式＞
＜実行文＞　：：＝　・　＜算術式＞　→　＜変数名＞
＜ if 文＞　：：＝　▲　＜論理式＞
＜反復文＞　：：＝　■　＜論理式＞
＜日本語文＞　：：＝　＜格の並び＞
＜格＞　：：＝　＜名詞句＞　＜助詞＞

図４　文法規則の例

### ２．４　主要ウィンドウ

ALEX2はウィンドウズプログラムであり、以下のようなウィンドウから構成される。

・メインウィンドウ

擬似言語プログラムの入力およびシステム全体の統括を行う。

・実行ウィンドウ

擬似言語プログラムを実行し、結果を表示する。

・入力ウィンドウ

データ入力を行う。

## ３．改訂概要

### ３．１　開発言語

ALEX1はMicrosoftのVisual C#で開発したが、ALEX2ではすべてをJava言語で書き換えた。これは、プラットフォームに依存しないJava言語の特長を生かすためである。Visual C#とJava言語はよく似た言語であるものの、細かな点で多くの違いがあり、変換作業はすべて手作業で行った。

また、ALEX2の開発には、フリーソフトであるeclipseを利用したが、ALEX2は実行可能JARファイルにしてあるので、すべてのパソコンで実行できる。

### ３．２　データ型

ALEX1で使用できるデータ型は整数型と実数型のみであったが、ALEX2では、以下の基本データ型が使用できるように改良した。

・整数型
・実数型
・文字型
・文字列型
・論理型

ALEX2では、すべての値をValueクラスで管理している。各種演算も

Valueクラスのメソッドで実現している。

**1）文字列型**

1. a）文字列定数

ALEX1では、文字列定数は

「面積＝」

のようにカギ括弧「と」で囲んで表していたが、一般的ではないので、ALEX2では、Java言語同様、

"面積＝"

のようにダブルクォートで囲むように変更した。

1. b）＋演算子

ALEX2では、文字列定数同士、もしくは、文字列定数と他のデータ型の値を＋演算子で連結して新たな文字列を作り出せるように改訂した。これは、ALEX1にはなかった機能である。＋演算子は代入文などの中で使用できる。

例えば、Ｘが整数型変数で、値が15.5のとき、代入文

・Ｙ←"面積は"＋Ｘ＋"である"

を実行すると、文字列型変数Ｙには

"面積は15.5である"

という文字列が代入される。

＋演算子は出力文でも用いることができるので、出力アルゴリズムを簡潔に表すことができる。

**2）論理型**

2. a）代入文

ALEX1では代入文は算術式に限定していたが、ALEX2では論理式の代入も可能にした。すなわち、

論理型変数　←　論理式

または

論理式　→　論理型変数

という代入式も受理する。

2. b）論理値を管理するクラス

ALEX1では選択構造や反復構造の条件式の値はBoolValueクラスで管理していたが、Valueクラスに論理型を組み込んだので、BoolValueクラスは削除した。

2. c）論理定数

論理定数はtrueとfalseのみである。「真」や「偽」という日本語表現も考慮したが、論理定数は論理式の代入文以外ではあまり使用することはないので、とりあえず、英語表記のみとした。

### ３．３　配列の添字

ALEX1では配列の添字は１からの連番としていたが、ALEX2ではＣ言語やJavaに合わせ、0からの連番とした。

### 3．4　組み込み関数

ALEX2では文字列型を追加したので、それに伴い、以下の関数を組み込んだ。

・length

文字列データを受け取り、その長さ（文字数）を返す。

・toCharArray

文字配列Ｘと文字列Ｙを引数として受け取り、Ｙの内容を１文字ずつバラバラにして配列Ｘに登録する。

・toString

文字列Ｘと文字配列Ｙを受け取り、配列Ｙの文字データをまとめて文字列Ｘに代入する。

### 3．5　構造体

基本情報技術者試験の中の擬似言語問題では、構造体が出題されたことはないが、リスト構造や木構造についての出題はあるので、構造体を利用した擬似言語プログラムも実行できるように、ALEX2を拡張した。なお、構造体を使ったプログラミングの問題は、応用情報技術者試験では出題されている。

#### 1）構造体の定義

構造体は、複数のデータの集まりである。構造体内のデータをメンバという。構造体を使用する場合には、事前に「定義」しておかなければならない。構造体の定義は図５の形式とする。

```
○構造体　構造体名：　{
    ○メンバ名の定義
        …
    ○メンバ名の定義
}
```

図５　構造体の定義形式

例えば、リスト構造のセル一つ分を表す構造体CELLは次のように記述する。

＜例１＞

```
○構造体　CELL：{
    ○整数型：data
    ○構造体　CELL：next
}
```

#### 2）構造体変数の定義

構造体を使用するには、さらに、構造体変数を定義しておかなければならない。構造体変数は図６のように定義する。

```
○構造体　構造体名：　変数名
```

図６　構造体の定義形式

例１のように、構造体の中に構造体変数を定義することもできる。これに

より、リスト構造や木構造などの複雑なデータ構造の処理が記述できる。

**3）構造体の要素**

構造体の要素を使用するには、

構造体変数名．メンバ名

のように、構造体変数名とメンバ名の間に「.」（ドット）を付ける。メンバが構造体変数の場合もあるので、

x.next.data

のような表記も受理する。

ドットの代わりに助詞「の」を使用することも検討したが、

x の next の data

のような表現はわかりやすいとは言えないので、採用しないこととした。

**4）null定数**

構造体変数の内容が空であることをnullで表す。null定数は、代入文や条件文の中で使用することができる。

＜例２＞

- x ← null

**5）関数値**

ALEX2では、関数の引数だけでなく、関数値として、構造体を返すことも可能とした。したがって、図７に示すような関数を定義することができる。

**6）文法規則**

構造体に関する記述を受理するための文法規則としては12個を追加した。そのうちのいくつかを図８に示す。

構造体を用いた簡単なサンプルプログラムの例を図９に示す。図９のプログラムを実行すると、１行目に１が、２行目に２が出力される。

```
○構造体　構造体名：　関数名（引数の並び）
            …
        関数の内容
            …
○
```

**図７　値として構造体を返す関数の枠組み**

```
＜構造体定義開始文＞：：＝○＜構造体ラベル＞＜識別子＞：{
＜構造体定義終了文＞：：＝}＜変数定義の並び＞
＜変数定義文＞：：＝○＜構造体ラベル＞＜識別子＞：＜変数定義の並び＞
＜構造体変数名＞：：＝＜変数名＞．＜変数名＞
＜構造体変数名＞：：＝＜構造体変数名＞．＜変数名＞
```

**図８　構造体に関する文法規則**

```
○構造体　ＬＩＳＴ:　{
  ○整数型：ｄａｔａ
  ○構造体　ＬＩＳＴ：ｎｅｘｔ
} ｐ、ｑ
・ｐ．ｄａｔａ←１
・ｐ．ｎｅｘｔ←ｑ
・ｑ．ｄａｔａ←２
・ｑ．ｎｅｘｔ←ｎｕｌｌ
・ｏｕｔｐｕｔ（ｐ）を呼び出す

○ｖｏｉｄ：ｏｕｔｐｕｔ（構造体　ＬＩＳＴ：ｘ）
■ｘ＜＞ｎｕｌｌ
  ・ｘ．ｄａｔａを出力する
  ・改行する
  ・ｘ←ｘ．ｎｅｘｔ
■
○
```

図９　構造体を使ったサンプルプログラム

なお、図９のプログラムの内部表現を図10に示す。なお、構造体におけるメンバ名の並びは図11のような木構造で表されるが、図10では、入力表現のまま、

p.data

のように表現している。

## ４．今後の課題

これまでALEXの改訂内容について述べてきたが、さらに、以下の点について改良を加える必要がある。

### １）複数データの書式付き入力の実現

現在のところ、データ入力は、１件ごととなっている。しかし、Ｃ言語には、複数データを一度に入力できる機能があり、入力の手間を省く上で、ALEXでも、同じ機能を実現すべきである。そのためには、「書式」の処理が必要となる。今後は、

「書式"%3d%2d"で整数値を変数xとyに入力する」

といった表現も受理できるよう改良する必要がある。

### ２）組み込み関数・述語の充実

既に、絶対値や平方根を求める関数、「等しい」や「大きい」、「偶数である」などの述語は処理できるように組み込

```
(構造体定義 LIST:
    (変数定義 整数型: data)
    (構造体変数定義 LIST: next)
)

関数定義数 = 2

(関数定義 void: main()
    (構造体変数定義 LIST: p q)
    (代入する p.data 1)
    (代入する p.next q)
    (代入する q.data 2)
    (代入する q.next null)
    (呼び出す (output p))
)

(関数定義 void: output(LIST:x)
    (while (≠ x null)
        (出力する x.data)
        (改行する)
        (代入する x x.next)
    )
)
```

**図10　図９のプログラムの内部表現**

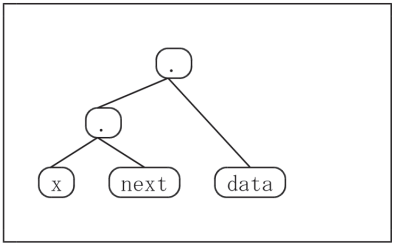


**図11　意味木の例**

んでいるが、さらにALEXを使いやすくするためには、組み込み関数や組み込み述語の充実も必要である。

ただし､「素数である」という述語や、「方程式の解」という関数などアルゴリズムを内包する表現については、処理できるだけでなく、擬似言語によるアルゴリズムの提示なども考慮する必要がある。

### ３）エラーメッセージ

ALEXのエラーメッセージは、簡単すぎる嫌いがある。ALEXが初心者向きの言語処理システムであることを考えると、エラーを修正しやすいようメッセージを工夫する必要がある。

特に、データ型のチェックの多くは、実行時に行っているが、中には構文解析時にチェックできる場合もある

ので、そのように変更する方が、システムの有用性は高まると思われる。

## 5．終わりに

ALEXはとりあえず実用に耐えうるレベルにまで高まったと思われるが、まだまだ課題も多い。今後もさらに改良を加え、プログラミングの初心者が楽に使用できるシステムを目指す必要がある。

## ＜参考文献＞

[1] Aho, Sethi, Ullman : Compilers, Addison Wesley (1986)

[2] 鑰山徹：初等的プログラミング教育のための擬似言語解析・実行システムALEXについて、パーソナルコンピュータ利用技術学会、Vol.2、No.1(2008)

[3] 掌田津耶乃：Eclipse4.3ではじめるJavaプログラミング入門、秀和システム (2013)

[4] 中田育男：コンパイラ、産業図書 (1989)

[5] 言語処理学事典、共立出版 (2009)

（かぎやま　とおる　本学教授）