構文解析とコンパイラ試験問題 2022 年度

2022 年 2 月 20 日

1　コンパイラとは

次の文章のそれぞれの四角の中に入る、**最も適切な語句１つ**を下のリストから選び、答えとしてその番号を書きなさい。

コンパイラとは高級言語などで記述された［　1　］を目的コード（オブジェクトコード）に変換するプログラムである。高級言語とは、人間に理解しやすい形式で書かれたプログラミング言語で、例えば［　2　］や、FORTRAN などがある。これらはコンパイラによって目的コード、例えば［　3　］に変換されるが人間がこれを読むのは難しい。

コンパイラは［　1　］を読み込むと、まず、字句解析を行い、［　4　］に分割する。そしてその後文法を解析する［　5　］を行い、目的コードを作成、出力する。

a. 状態遷移図　b. オートマトン　c. 機械語　d. C 言語
e. 原始プログラム（ソースプログラム）　f. トークン　g. 構文解析　h. CPU

2　正規表現とオートマトン 1

次の正規表現からオートマトンを作成せよ。終了状態は二重丸で示せ。

$$b(a|b) * ab$$

3　正規表現とオートマトン 2

C 言語（に似た言語）を字句解析するためのプログラムを考える。この言語は次の表に示す６つのトークンをもつ。表の穴①, ② をうめよ

トークンの種類	正規表現	備考
キーワード (for)	for	for
整数	[0-9][0-9]*	
浮動小数点数	①	実数　小数点が必要
識別子	②	英文字と_で始まり、英文字と_と数字が繰り返される長さ 1 以上の長さの文字列
区切り	スペース, セミコロン, カンマ	区切りをあらわす
OPERATOR（演算子）	+,=	四則演算

4 文法の定義

四則演算を表す文法 G1 を次に示す。これについて各問いに答えよ。(定義中の **num** は数字（[0-9][0-9]*）を示す）この文法は演算の優先順位は規定していない。優先順位は考えなくて良い。

$$
\begin{aligned}
G1 = (\quad & N \quad , T\ ,\ P\ ,\ S\ ) \\
N = \{\quad & E \quad \} \\
T = \{\quad & + \quad , \ - \ , \ * \ , \ / \ , \ \mathbf{num}\ \} \\
P = \{\quad & E \quad \to E\ +\ E \\
& E \quad \to E\ -\ E \\
& E \quad \to E\ *\ E \\
& E \quad \to E\ /\ E \\
& E \quad \to \mathbf{num}\} \\
S = \{\quad & E \quad \}
\end{aligned}
$$

1. 上の文法の構文図を示せ。終端記号は丸で囲み、非終端記号は四角で囲って示せ。
2. シフト還元構文解析を行う様子を順を追ってかけ。下に示すようにスタックと入力文字の組み合わせで書くこと

スタック		入力文字列
ϕ	$\leftarrow$	$5*6/2*8$

3. この文法は、一般的に数式に使用される () を使うことができない。() を使えるように文法を改変せよ。どこをどう変える、なにを加える、などの方法で、変更箇所だけ示せばよい。

5　文法 2（応用問題）

次に示す文法について、問いに答えよ。

$$
\begin{aligned}
G1 = (\ & N\ ,\ T\ ,\ P\ ,\ S\ ) \\
N = \{\ & E\ ,\ T\ ,\ F\ \} \\
T = \{\ & +\ ,\ -\ ,\ *\ ,\ /\ ,\ \mathbf{num}\ \} \\
P = \{\ & E \rightarrow E\ +\ T \\
& E \rightarrow E\ -\ T \\
& E \rightarrow T \\
& T \rightarrow T\ *\ F \\
& T \rightarrow T\ /\ F \\
& T \rightarrow F \\
& F \rightarrow \mathbf{num}\} \\
S = \{\ & E\ \}
\end{aligned}
$$

この文法に基づいて、次の文の最左導出を示せ。あわせて解析木を示せ

1. 3+4*5+2

6 構文解析

次のような yacc プログラム (yacc1.y) を作成した。なお、行頭の数字は行番号である。このプログラムと共に使用する lex(lex1.l) は参考のために最後に示すので必要に応じて参照せよ。

(yacc1.y) C 言語風構文解析プログラム

```
 1 %{
 2 #include <stdio.h>
 3 %}
 4
 5   %token ID                /* 変数を示す */
 6   %token NUM               /* 数字を示す   */
 7   %token IF ELSE           /* "if" "else" */
 8   %token WHILE             /* "while"   */
 9   %token LP RP             /* "(" ")"   */
10   %token LC RC             /* "{" "}"   */
11   %token SC                /* ";"       */
12   %token ASSIGN            /* "="       */
13   %token EQ NE             /* "==" "!=" */
14   %token FOR               /* "for"     */
15   %token DO                /* "do"      */
16   %token ELSE              /* "else"    */
17   %left  ADD               /* "+"       */
18   %left  SUB               /* "-"       */
19   %%
20   stmt :  ifstmt | assignstmt | whilestmt
21   ;
22
23   ifstmt : IF LP condE RP stmt |IF LP condE RP stmt ELSE stmt;
24
25
26   assignstmt : ID ASSIGN E SC
27   ;
28
29    whilestmt : WHILE LP condE RP stmt
30    ;
31
32   condE : E EQ E |E NE E
33   ;
34
35
36   E : E ADD E
37   | E SUB E
38   | ID
39   | NUM
```

```
40    ;
41    %%
```

1. yacc1.y で定義されている非終端記号を全て示せ
2. このプログラムの開始状態となる非終端記号は何か？
3. このプログラムでは、一文（一行）から構成されるプログラムしか入力として受け付けない。改造して複数の文を入力として受け付けることができるようにせよ。（〜を、何行目に入れる、という形式で答えるとよい）
4. 次に示す（例 1）のように、while 文内で複数の文を扱うために、**{ }** を導入して、複合文を扱えるようにプログラムを改変せよ。

   （例 1）
   ```
   while(i != 10){
     x = x +y;
     i = i+1;
   }
   ```

5. 23 行目に示されている文法について、構文図で示せ。なお、終端記号は丸で囲み、非終端記号は四角で囲んで表記せよ。
6. つぎのような for 文を解析できるようにしたい。

   （例 2）オリジナルの `for` 文
   ```
   for(x = 0; x != 10; x = x+1;)
     a=b+c;
   ```

   （簡単のため）for() のカッコ内の最後のところに ; が付いている。この文を解析できるように yacc プログラムを改良せよ。（追加プログラムを示して、それが何行目に入るか示す）
7. 次のプログラムが入力されたときに生成される解析木を書け

   ```
   if(y!=0)
     d=3;
   ```

(lex1.l) yacc1.y 用の lex プログラム

```
 1 %{
 2 # include "y.tab.h"
 3 extern int yylval;
 4 int line = 1;
 5 %}
 6
 7 %%
 8 "+"   {return ADD;}
 9 "-"   {return SUB;}
10 "="   {return ASSIGN;}
11 "=="  {return EQ;}
12 "!="  {return NE;}
13 "{"   {return LC;}
14 "}"   {return RC;}
15 "("   {return LP;}
16 ")"   {return RP;}
17 ";"   {return SC;}
18 "if"      {return IF;}
19 "else"    {return ELSE;}
20 "while"   {return WHILE;}
21 "FOR"     {return FOR;}
22 "do"      {return DO;}
23 [0-9]+ {  yylval = atoi(yytext); return NUM;}
24 [a-zA-Z] {  yylval = yytext[0];  return ID;}
25
26 "\n" {printf("line %d:\n",line++);}
27 [ \t] ;
28
29 %%
30
31 main(){
32     return yyparse();
33 }
34
35 yyerror(char *s){
36   printf("%s\n",s);
37 }
```