情報科学実験Ｂ

担当：松本 真佑 先生
提出者：秋口　敬
コース：計算機科学コース
学籍番号：09B19002

提出年月日：2021 年 11 月 19 日

1　課題の目的

Pascal 風言語で記述されたプログラムのトークン列ファイルを入力として，プログラムが構文的に正しいかどうか判定するプログラムを作成する．

2　課題達成の方針と設計

2.1　方針

LL(1) 解析を用いて構文解析を行う。

2.2　設計

初めに ts ファイルを一行ずつ読み取り、文字列のみのリスト、トークン名のみのリスト、トークン ID のみのリスト、行番号のみのリストとなっている配列を作成する。
トークンを一つずつみていき、それが前のトークンの EBNF 構文と合致しているか確認する。
終端記号の場合は、合致していたら次のトークンを見ていく。非終端記号の場合は、そのトークンの EBNF 構文について正しいかどうか確認する。これを終端記号になるまで繰り返す。
これを繰り返す。
もし EBNF 構文に合致しないトークンが出てきたなら、そのトークンがある行番号を出力する。
以降はエラーがあっても出力は行わないようにする。

3　プログラムの実装方針

ここでは主に、LL(1) 解析を用いた、EBNF 構文と合致しているかどうかの確認の処理について説明する。

3.1　基本的な EBNF 構文との合致判定

ここでは、「プログラム ＝ ”program” プログラム名 ”;” ブロック 複合文 ”.”.」の構文解析について説明する。
まず、終端記号である program という文字列が来なければエラー出力をする。その次は非終端記号「プログラム名」が来る必要があるので、別メソッドでプログラム名についての EBNF 構文との合致判定を行う。その次は終端記号である「;」が来なければエラー、非終端記号のブロックについて別メソッドで判定、非終端記号の複合文について別メソッドで判定、終端記号である「.」が来なければエラー、となっている。

```
void program() {
if (!(jiku[n].equals("program"))) {
if(err==0) {
System.err.println("Syntax error: line "+gyou[n]);
}
```

```
err++;
}
n++;

programName();
if (!(jiku[n].equals(";"))) {
if(err==0) {
System.err.println("Syntax error: line "+gyou[n]);
}
err++; // プログラム名の次に ";" が出てこなければ構文エラー
}
n++;
block();             // 別 EBNF メソッド
complexStatement(); // 別 EBNF メソッド

if (!(jiku[n].equals("."))) {
if(err==0) {
System.err.println("Syntax error: line "+gyou[n]);
}
err++;
}
n++;

}
```

3.2 選択が任意であるフレーズがある EBNF 構文との合致判定

ここでは、「手続き呼出し文 = 手続き名 [”(” 式の並び ”)”].」の構文解析について説明する。まず、非終端記号の手続き名の合致判定を行う。次に、もし「(」がきたら、「 ”(” 式の並び ”)” 」が来るとし、その後は式の並び、「)」との合致判定を行う。もし「(」が来ない場合は「 ”(” 式の並び ”)” 」はこないので次の合致判定に移る。

```
void tetudukiyobidasi() {
if(!tokenname[n].equals("SIDENTIFIER")) {
if(err==0) {
System.err.println("Syntax error: line "+gyou[n]);
}
err++;
}
n++;
if(jiku[n].equals("(")) {
n++;
sikinonarabi();
```

```
if (!(jiku[n].equals(")"))) {
if(err==0) {
System.err.println("Syntax error: line "+gyou[n]);
}
err++;
}
n++;
}

}
```

3.3 繰り返してよいフレーズがある EBNF 構文との合致判定

ここでは、「式の並び = 式 ”,” 式 」の構文解析について説明する。「 ”,” 式」が繰り返してよいフレーズである。
まず、式の合致判定を行う。次に、もし「,」があれば、ループに入る。ループの中では「,」と式との合致判定を行います。次のトークンが「,」でなければループから抜ける。

```
void sikinonarabi() {
siki();
if(jiku[n].equals(",")) {
while(true) {
if(!(jiku[n].equals(","))){
if(err==0) {
System.err.println("Syntax error: line "+gyou[n]);
}
err++;
}
n++;
siki();

if(!(jiku[n].equals(","))){
break;
}
}
}
}
```

4 まとめ

EBNF 構文との合致判定に関してはスムーズに作業することができた。構文木の作成にも取り掛かったが、このままのメソッドで構文木を作る良い案が思いつかず、次回以降に回すことにした。

テストファイルではうまくいっているが、実際には間違っている、というパターンがあるかもしれないので、次回以降の課題はそのことも頭に入れつつ作業を進めていきたい。

5　感想、謝辞

課題をいくつかのステップに分けて行うことでスムーズにプログラムを作成することかできた。授業や slack での質問に答えていただいた教員の方々に感謝の意を表します。