レポート

システムソフトウェア特論 課題#9

担当教員 久野靖

提出日 2020 年 2 月 17 日

電気通信大学
情報理工学研究科
情報・ネットワーク工学専攻

学籍番号 1931010

伊勢大平

1　課題

選んだ演習問題は 9-3 である．
第 2 節 (課題の再掲) で，課題を要約して説明する．

2　課題の再掲

SableCC を用いて自分の好きな言語を設計して実装する．

3 方針

3.1 SableCC プログラムについて

設計した言語を BNF で記述し，それを元に SableCC プログラムを作ればよい．
端記号と非端記号とを区別するため，端記号は山括弧 (<>) で囲って表記する．
端記号に対応する意味は次の通りである．

- $< prog >$ ⋯ プログラム
- $< stlist >$ ⋯ 文の列
- $< stat >$ ⋯ 文
- $< term >$ ⋯ 項（識別子またはリテラルであるもの）
- $< ident >$ ⋯ 識別子
- $< iconst >$ ⋯ リテラル（数値リテラル（整数のみ））

設計した言語を BNF で記述すると次のようになる．

- $< prog > ::= < stlist >$
- $< stlist > ::= < stlist >< stat > \mid \epsilon$
-

$$
\begin{aligned}
< stat > ::= & \ assign \ < ident >, < term > ; \\
& \mid read \ < ident > ; \\
& \mid print \ < term > ; \\
& \mid add \ < ident >, < term >, < term > ; \\
& \mid sub \ < ident >, < term >, < term > ; \\
& \mid lt \ < ident >, < term >, < term > ; \\
& \mid gt \ < ident >, < term >, < term > ; \\
& \mid eq \ < ident >, < term >, < term > ; \\
& \mid if \ (< ident >), \{< stlist >\} ; \\
& \mid while \ (< ident >), \{< stlist >\} ; \\
& \mid dowhile \ (< ident >), \{< stlist >\} ;
\end{aligned}
$$

- $< term > ::= < ident > \mid < iconst >$

3.2 Java(Executor) プログラムについて

次のような言語の決まりに基づいて作成する．

- 「assign a,b;」… b の値が a に格納される．a は識別子であり，b は識別子でも数値リテラルでもよい．
- 「read a;」… ターミナルに「a>」を表示し，ターミナルから入力された数値を a に格納する．a は識別子である．
- 「add a,b,c;」… b+c の値が a に格納される．a は識別子であり，b，c は識別子でも数値リテラルでもよい．
- 「sub a,b,c;」… b-c の値が a に格納される．a は識別子であり，b，c は識別子でも数値リテラルでもよい．
- 「lt a,b,c;」… b<c であれば a に 1 が格納され，そうでなれけば a に 0 が格納される．a は識別子であり，b，c は識別子でも数値リテラルでもよい．
- 「gt a,b,c;」… b>c であれば a に 1 が格納され，そうでなれけば a に 0 が格納される．a は識別子であり，b，c は識別子でも数値リテラルでもよい．
- 「eq a,b,c;」… b=c であれば a に 1 が格納され，そうでなれけば a に 0 が格納される．a は識別子であり，b，c は識別子でも数値リテラルでもよい．
- 「if (a),{ b };」… a の値が 1 であれば b を実行する．a は識別子であり，b は文の列である．
- 「while (a),{ b };」… a の値が 1 である間は b を繰り返し実行する．a は識別子であり，b は文の列である．
- 「dowhile (a),{ b };」… はじめに b を一回実行し，その後は，a の値が 1 である間は b を繰り返し実行する．a は識別子であり，b は文の列である．

4 成果物

4.1 プログラム

SableCC のプログラムはリスト 1 のように書いた.

リスト 1: SableCC のソースコード (minLang.sablecc)

```
Package minLang;

Helpers
  digit = ['0'..'9'] ;
  lcase = ['a'..'z'] ;
  ucase = ['A'..'Z'] ;
  letter = lcase | ucase ;

Tokens
  iconst = ('+'|'-'|) digit+ ;
  blank = (' '|13|10)+ ;
  read = 'read' ;
  print = 'print' ;
  semi = ';' ;
  comma = ',' ;
  assign = 'assign';
  add = 'add' ;
  sub = 'sub' ;
  lt = 'lt' ;
  gt = 'gt' ;
  eq = 'eq' ;
  if = 'if' ;
  while = 'while' ;
  dowhile = 'dowhile' ;
  lbra = '{' ;
  rbra = '}' ;
  lpar = '(' ;
  rpar = ')' ;
  ident = letter (letter|digit)* ;

Ignored Tokens
  blank ;

Productions
  prog = {stlist} stlist
       ;
  stlist = {stat} stlist stat
         | {empty}
         ;
  stat = {assign} assign ident comma term semi
       | {read} read ident semi
       | {print} print term semi
       | {add} add [left]:ident [comma1]:comma [mid]:term [comma2]:comma [right]:term semi
       | {sub} sub [left]:ident [comma1]:comma [mid]:term [comma2]:comma [right]:term semi
       | {lt} lt [left]:ident [comma1]:comma [mid]:term [comma2]:comma [right]:term semi
       | {gt} gt [left]:ident [comma1]:comma [mid]:term [comma2]:comma [right]:term semi
       | {eq} eq [left]:ident [comma1]:comma [mid]:term [comma2]:comma [right]:term semi
       | {if} if lpar [bool]:ident rpar comma lbra stlist rbra semi
       | {while} while lpar [bool]:ident rpar comma lbra stlist rbra semi
       | {dowhile} dowhile lpar [bool]:ident rpar comma lbra stlist rbra semi
       ;
  term = {ident} ident
       | {iconst} iconst
       ;
```

Java(MinLang) のプログラム (トップのプログラム) はリスト 2 のように書いた.

リスト 2: Java(MinLang) のソースコード (MinLang.java)

```
1  package minLang;
2  import minLang.parser.*;
3  import minLang.lexer.*;
4  import minLang.node.*;
5  import java.io.*;
6  import java.util.*;
7
8  public class MinLang {
9      public static void main(String[] args) throws Exception {
10         Parser p = new Parser(new Lexer(new PushbackReader(
11                                                 new InputStreamReader(new
                                                       FileInputStream(args[0]))
                                                       ,1024)));
12         Start tree = p.parse();
13         tree.apply(new Executor());
14     }
15 }
```

Java(Executor) のプログラムはリスト 3 のように書いた.

リスト 3: Java(Executor) のソースコード (Executor.java)

```
1  package minLang;
2  import minLang.analysis.*;
3  import minLang.node.*;
4  import java.io.*;
5  import java.util.*;
6
7  class Executor extends DepthFirstAdapter {
8      Scanner sc = new Scanner(System.in);
9      PrintStream pr = System.out;
10     HashMap<String,Integer> vars = new HashMap<String,Integer>();
11
12     // ----- term ----- //
13     // term = {ident} ident
14     @Override
15     public void outAIdentTerm(AIdentTerm node) {
16         String s = node.getIdent().getText();
17         if(!vars.containsKey(s)) vars.put(s, new Integer(0));
18         setOut(node, vars.get(s));
19     }
20     // term = {iconst} iconst
21     @Override
22     public void outAIconstTerm(AIconstTerm node) {
23         setOut(node, new Integer(node.getIconst().getText()));
24     }
25
26     // ----- stat ----- //
27     // stat = {assign} assign ident comma term semi
28     @Override
29     public void outAAssignStat(AAssignStat node) {
30         String s = node.getIdent().getText();
31         vars.put(s, (Integer)getOut(node.getTerm()));
32     }
33     // stat = {read} read ident semi
34     @Override
35     public void outAReadStat(AReadStat node) {
36         String s = node.getIdent().getText();
37         pr.print(s + "> ");
38         vars.put(s, sc.nextInt()); sc.nextLine();
39     }
40     // stat = {print} print term semi
41     @Override
```

```
42      public void outAPrintStat(APrintStat node) {
43          pr.println(getOut(node.getTerm()));
44      }
45      // stat = {add} add [left]:ident [comma1]:comma [mid]:term [comma2]:comma [right]:
            term semi
46      @Override
47      public void outAAddStat(AAddStat node) {
48          String s = node.getLeft().getText();
49          int v = (Integer)getOut(node.getMid()) + (Integer)getOut(node.getRight());
50          vars.put(s, v);
51      }
52      // stat = {sub} sub [left]:ident [comma1]:comma [mid]:term [comma2]:comma [right]:
            term semi
53      @Override
54      public void outASubStat(ASubStat node) {
55          String s = node.getLeft().getText();
56          int v = (Integer)getOut(node.getMid()) - (Integer)getOut(node.getRight());
57          vars.put(s, v);
58      }
59      // stat = {lt} lt [left]:ident [comma1]:comma [mid]:term [comma2]:comma [right]:
            term semi
60      @Override
61      public void outALtStat(ALtStat node) {
62          String s = node.getLeft().getText();
63          int v = ((Integer)getOut(node.getMid()) < (Integer)getOut(node.getRight())) ? 1
                : 0;
64          vars.put(s, v);
65      }
66      // stat = {gt} gt [left]:ident [comma1]:comma [mid]:term [comma2]:comma [right]:
            term semi
67      @Override
68      public void outAGtStat(AGtStat node) {
69          String s = node.getLeft().getText();
70          int v = ((Integer)getOut(node.getMid()) > (Integer)getOut(node.getRight())) ? 1
                : 0;
71          vars.put(s, v);
72      }
73      // stat = {eq} eq [left]:ident [comma1]:comma [mid]:term [comma2]:comma [right]:
            term semi
74      @Override
75      public void outAEqStat(AEqStat node) {
76          String s = node.getLeft().getText();
77          int v = ((Integer)getOut(node.getMid()) == (Integer)getOut(node.getRight())) ?
                1 : 0;
78          vars.put(s, v);
79      }
80      // stat = {if} if lpar [bool]:ident rpar comma lbra stlist rbra
81      @Override
82      public void caseAIfStat(AIfStat node){
83          String s = node.getBool().getText();
84          if(vars.get(s)==1) { node.getStlist().apply(this); }
85      }
86      // stat = {while} while lpar [bool]:ident rpar comma lbra stlist rbra semi
87      @Override
88      public void caseAWhileStat(AWhileStat node){
89          String s = node.getBool().getText();
90          while(true){
91              if(vars.get(s)==1) { node.getStlist().apply(this); }
92              else { return; }
93          }
94      }
95      // stat = {dowhile} dowhile lpar [bool]:ident rpar comma lbra stlist rbra semi
96      @Override
97      public void outADowhileStat(ADowhileStat node){
98          String s = node.getBool().getText();
```

```
99          while(true){
100             if(vars.get(s)==1) { node.getStlist().apply(this); }
101             else { return; }
102         }
103     }
104
105 }
```

実行プログラムは二つ作成した．リスト 4 とリスト 5 のように書いた．

リスト 4: 実行用のソースコード 1(fib1.min)

```
1  read a;
2  read b;
3
4  lt bool,a,b;
5  if (bool),{ sub n,b,a; };
6  eq bool,a,b;
7  if (bool),{ assign n,0; };
8  gt bool,a,b;
9  if (bool),{ sub n,a,b; };
10
11 assign x,0;
12 assign y,1;
13
14 dowhile (bool),{
15   add sum,x,y;
16   lt bool,sum,n;
17   if (bool),{ print sum; };
18   assign x,y;
19   assign y,sum;
20 };
```

リスト 5: 実行用のソースコード 2(fib2.min)

```
1  read a;
2  read b;
3
4  lt bool,a,b;
5  if (bool),{ sub n,b,a; };
6  eq bool,a,b;
7  if (bool),{ assign n,0; };
8  gt bool,a,b;
9  if (bool),{ sub n,a,b; };
10
11 assign x,0;
12 assign y,1;
13
14 assign bool,1;
15 while (bool),{
16   add sum,x,y;
17   lt bool,sum,n;
18   if (bool),{ print sum; };
19   assign x,y;
20   assign y,sum;
21 };
```

4.2 プログラムの説明

4.2.1 SableCC プログラムについて

SableCC プログラムは第 3.1 節で述べた BNF に従って Productions を書いた．「add a,b,c;」のような命令は，a は ident であり，b と c は term である．b と c の term は区別するため名前を付ける必要がある．a は区別の必要は無いが，Java での記述が分かりやすくなるように，a を left，b を mid，c を right のように名前を付けた．なお，二つでてくるコンマは区別する意味は無いがプログラム上区別しないといけないので名前を付けた．

4.2.2 Java プログラムについて

オーバライドしたメソッドについては第 3.2 で述べた決まりに従って書いた．それぞれのメソッドについての説明は以下の通りである．

outAIdentTerm 識別子 ident の名前を getText() で取得し s に入れる．s という名前の識別子があれば，その識別子の値を vars.get(s) で取得し，上側のノードに向かってデータを戻せるように setOut する．

outAIconstTerm 数値リテラル iconst の文字列を取得したら，それを数値に変換して setOut する．

outAAssignStat 識別子 ident の名前を getText() で取得し s に入れる．識別子の名前 s と term の値をペアにして，表 vars に格納する．

outAReadStat 識別子 ident の名前を getText() で取得し s に入れる．識別子の名前 s とターミナルから入力された数値をペアにして，表 vars に格納する．

outAPrintStat term の値を getOut で取得し，println で表示する．

outAAddStat 識別子 ident の名前を getText() で取得し s に入れる．mid(term) の値と right(term) の値を getOut で取得し，その和を v とする．識別子の名前 s と値 v をペアにして，表 vars に格納する．なお，言語の仕様上，上側のノードに向かってデータを戻す必要はないので，setOut は不要である．

outASubStat 識別子 ident の名前を getText() で取得し s に入れる．mid(term) の値と right(term) の値を getOut で取得し，その差を v とする．識別子の名前 s と値 v をペアにして，表 vars に格納する．

outALtStat 識別子 ident の名前を getText() で取得し s に入れる．mid(term) の値と right(term) の値を getOut で取得し，「mid の値 ＜right の値」であれば v を 1 とし，そうでなければ v を 0 とする．識別子の名前 s と値 v をペアにして，表 vars に格納する．

outAGtStat 識別子 ident の名前を getText() で取得し s に入れる．mid(term) の値と right(term) の値を getOut で取得し，「mid の値 >right の値」であれば v を 1 とし，そうでなければ v を 0 とする．識別子の名前 s と値 v をペアにして，表 vars に格納する．

outAEqStat 識別子 ident の名前を getText() で取得し s に入れる．mid(term) の値と right(term) の値を getOut で取得し，「mid の値 =right の値」であれば v を 1 とし，そうでなければ v を 0 とする．識別子の名前 s と値 v をペアにして，表 vars に格納する．

caseAIfStat stlist が実行されるかされないかは bool の値次第である．よって，途中処理が必要となり，out ではなく case メソッドを用いる．bool(ident) の名前を getText() で取得し s に入れる．s とペアになってる値（意味的には識別子 ident の値）が 1 であれば stlist を実行する．

caseAWhileStat stlist が実行されるかされないかは bool の値次第である．よって，途中処理が必要となり，out ではなく case メソッドを用いる．bool(ident) の名前を getText() で取得し s に入れる．s とペアになってる値（意味的には識別子 ident の値）が 1 である間は stlist を繰り返し実行する．

outADowhileStat while と異なり，必ず一回は stlist が実行される．よって，out メソッドでよい．ノードをたどった際に一回 stlist が実行されるので，ノードから出て行くときに while と同様の動作をすればよい．よって，メソッドの中身の記述は caseAWhileStat と同じでよい．

4.2.3 テストプログラム (fib1.min と fib2.min) について

fib1.min（リスト 4）では，1 行目で a の値をターミナルから入力された値とし，次に 2 行目で b の値をターミナルから入力された値とする．4 から 9 行目により，a と b の差の絶対値が n に代入される．11 行目から 20 行目により，n までのフィボナッチ数が表示される．

fib2.min（リスト 5）は fib1.min と同様の動作をする．fib1.min では dowhile を用いたが，fib2.min では while を用いている．8 行目で bool が 1 になっていない場合を考えて，while の前に bool の値を 1 にしておく必要がある．

4.3 実行例

プログラムの実行結果はリスト 6,7 のようになった．

リスト 6 ではリスト 4 の実行結果を 3 通り示し，リスト 7 ではリスト 5 の実行結果を 3 通り示している．

リスト 6 で 1 行目の実行では a に 50 を，b に 90 を代入している．$|50 - 90| = 40$ までのフィボナッチ数が表示されている．12 行目の実行では a に 90 を，b に 50 を代入している．$|90 - 50| = 40$ までのフィボナッチ数が表示されている．23 行目の実行では a に 70

を，b に 70 を代入している．$|70-70|=0$ なので，1 つもフィボナッチ数が表示されていない．以上より，fib1.min は正しく動作していると分かる．

同様にして，リスト 7 から，fib2.min が正しく動作していると分かる．

fib1.min と fib2.min が正しく動作していることから，全ての命令 (assign，read，add，sub，lt，gt，eq，if，while，dowhile) が正しく動作していると分かる．

リスト 6: 実行結果 1(fib1.min の実行結果)

```
$ java minLang.MinLang fib1.min
a> 50
b> 90
1
2
3
5
8
13
21
34
$ java minLang.MinLang fib1.min
a> 90
b> 50
1
2
3
5
8
13
21
34
$ java minLang.MinLang fib1.min
a> 70
b> 70
```

リスト 7: 実行結果 2(fib2.min の実行結果)

```
$ java minLang.MinLang fib2.min
a> 50
b> 90
1
2
3
5
8
13
21
34
$ java minLang.MinLang fib2.min
a> 90
b> 50
1
2
3
5
8
13
21
```

```
34
$ java minLang.MinLang fib2.min
a> 70
b> 70
```

5　考察

if 文，while 文，dowhile 文を定義する代わりに，ジャンプ文を定義したかったが，どうすればできるのかが分からなかった．考えたこととしては，ラベルを導入して「stlist = label stlist」のような production と「stat = jmp label semi」のような production を追加し,「jmp L1;」という文を実行する際，そのまま jmp 文のノードをたどって出て行くのではなく，たどる場所を L1 の親ノード（stlist ノード）に変えるという方法である．しかし，どうすれば実装できるかが分からなかった．

6　アンケートの解答

Q1 SableCC のようなコンパイラコンパイラについて，使ってみてどのように思いましたか．

A1 使い方がシンプルで，使いやすかったです．Cup と JFlex を連携して使う場合と，SableCC を使う場合で，どっちの方が応用が効くのか疑問に思いました．

Q2 Visitor パターンについて知っていましたか．使ってみてどのように思いましたか．

A2 Visitor パターンがどのようなものか詳しくは知っていませんでした．使ってみて，Visitor パターンがどう活かされているのか，いまいち実感できませんでした．

Q3 リフレクション（課題をやってみて気づいたこと），感想，要望など．

A3 自分で設計した言語が期待通りに動いたときは嬉しかったです．

プログラムがかけても，プログラムの説明を書くのは言葉が上手く出てこなくて大変でした．適切な説明をするために，どのような用語，表現を使えばいいのか考えるのが難しかったです．