2015年度 第3回 言語処理系レポート課題

課題 1 から課題 6 までに取り組み, レポートを提出しなさい.

課題 1

以下の do-while 文を PL/0' 言語に加えるために, 以下の課題に取り組みなさい.

生成規則 *statement* → **do** *statement* **while** *condition*

動作 **do** *statement* **while** *condition* の動作を次のように定義する.

1. *statement* を実行する.
2. *condition* の評価値が真なら 1 へ. 偽ならば繰り返しを終了する.

課題 1-1

compile.c の isStBeginKey 関数を変更し, 字句 `do` を非終端記号 *statement* の先頭記号の集合に加えなさい. レポートには, コンパイラに加えた変更点について書きなさい.

課題 1-2

compile.c の statement 関数を変更し, do-while 文に対応する目的コード (図 1) を出力するようにしなさい. レポートには, コンパイラに加えた変更点について書きなさい.

```
label1:   statement のコード
          condition のコード
          jpc label2
          jmp label1
label2:
```

図 1: do-while 文に対応する目的コード

課題 1-3

do-while 文の動作を確認するための PL/0' プログラム do.pl0(図 2) を実行しなさい。レポートには, do.pl0 の動作結果 (出力) を掲載しなさい.

```
var x;
begin
   x := 0;
   do begin
      write x;
      writeln;
      x := x + 1
   end
   while x < 3
end.
```

図 2: 動作確認プログラム do.pl0

課題 2

以下の repeat-until 文を PL/0' 言語に加えるために, 以下の課題に取り組みなさい.

生成規則 *statement* → **repeat** *statement* **until** *condition*

動作 **repeat** *statement* **until** *condition* の動作を次のように定義する.

1. *statement* を実行する.
2. *condition* が偽なら 1 へ. 真ならば繰り返しを終了する.

課題 2-1

do-while 文の目的コード (図 1) を参考にして, repeat-until 文に対応する目的コードを書きなさい.

課題 2-2

getSource.h, getSource.c を変更して, 字句 `repeat` と字句 `until` を登録しなさい. レポートには, コンパイラに加えた変更点について書きなさい.

課題 2-3

compile.c の isStBeginKey 関数を変更し, 字句 `repeat` を非終端記号 *statement* の先頭記号の集合に加えなさい. レポートには, コンパイラに加えた変更点について書きなさい.

課題 2-4

compile.c の statement 関数を変更し, repeat-until 文に対応する目的コードを出力するようにしなさい. レポートには, コンパイラに加えた変更点について書きなさい.

課題 2-5

repeat-until 文の動作を確認するための PL/0' プログラム repeat.pl0(図 3) を実行しなさい。レポートには, repeat.pl0 の動作結果 (出力) を掲載しなさい.

```
var x;
begin
   x := 0;
   repeat begin
      write x;
      writeln;
      x := x + 1
   end
   until x=3
end.
```

図 3: 動作確認プログラム repeat.pl0

課題 3

以下の if-then-else 文を PL/0' 言語に加えるために, 以下の課題に取り組みなさい.

生成規則 $\mathit{statement} \rightarrow \textbf{if}\ \mathit{condition}\ \textbf{then}\ \mathit{statement}_1\ (\textbf{else}\ \mathit{statement}_2 \mid \epsilon)$

動作 **if** *condition* **then** $\mathit{statement}_1$ (**else** $\mathit{statement}_2 \mid \epsilon$) の動作を次のように定義する.

1. *condition* を評価する.
2. *condition* の評価結果が真ならば, $\mathit{statement}_1$ を実行する.
3. *condition* の評価結果が偽かつ $\mathit{statement}_2$ があれば, $\mathit{statement}_2$ を実行する.

補足 言語のあいまい性を解決するために, 「**else** を見たとき, すでに現れた **then** の中で, まだどの **else** とも組み合わされていないもので, その **else** に最も近いものと組み合わせる. 」とする. (参考:教科書 P99. 問題 5.4)

課題 3-1

do-while 文の目的コード (図 1) を参考にして, **if** *condition* **then** $\mathit{statement}_1$ **else** $\mathit{statement}_2$ に対応する目的コードを書きなさい.

課題 3-2

getSource.h, getSource.c を変更して, 字句 `else` を登録しなさい. レポートには, コンパイラに加えた変更点について書きなさい.

課題 3-3

compile.c の statement 関数を変更し, if-then-else 文に対応する目的コードを出力するようにしなさい. レポートには, コンパイラに加えた変更点について書きなさい.

課題 3-4

if-then-else 文の動作を確認するための PL/0' プログラム else.pl0(図 4) を実行しなさい。レポートには, else.pl0 の動作結果 (出力) を掲載しなさい.

```
var x;
begin
   x := 0;
   while x<3 do begin
      if x < 1 then write 0
      else if x < 2 then write 1
      else write 2;
      writeln;
      x := x+1;
   end;
end.
```

図 4: 動作確認プログラム else.pl0

課題 4

PL/0' 言語に配列を導入しなさい.

課題 4-1

配列を使うために, PL/0' 言語に加える文法の変更点を述べよ. (配列の宣言と参照, 配列への代入について考慮しなさい.)

課題 4-2

配列に対応するために, PL/0' 仮想機械に新たな命令語が必要となるかを検討しなさい. もし必要ならば, それらの各命令語について, 以下のことを述べなさい.

1. 自分で決めたニーモニック
2. その命令の動作

課題 4-3

配列に対応するように, PL/0' コンパイラを変更しなさい. レポートには, コンパイラに加えた変更点について書きなさい.

課題 4-4

配列の導入を確認するための PL/0' プログラム array.pl0 を作成しなさい. レポートには, array.pl0 とその動作結果を掲載しなさい. (注意: array.pl0 はなるべく簡単なプログラムにすること.)

課題 5

PL/0' 言語に手続き (戻り値がない関数) を導入しなさい. 手続きの呼び出し方は,

call 手続き名 (引数 1, 引数 2, ..., 引数 n)

という文で行うことにする.

課題 5-1

手続きを導入するために, PL/0' 言語に加える文法の変更点を述べよ. (手続きの宣言と呼び出し方について考慮しなさい.)

課題 5-2

手続きに対応するためには, PL/0' 仮想機械に新たな命令語が必要となるかを検討しなさい. もし必要ならば, それらの各命令語について, 以下のことを述べよ.

1. 自分で決めたニーモニック
2. その命令の動作

課題 5-3

手続きに対応するように, PL/0' コンパイラを変更しなさい. レポートには, コンパイラに加えた変更点について書きなさい.

課題 5-4

手続きの導入を確認するための PL/0' プログラム proc.pl0 を作成しなさい. レポートには, proc.pl0 とその動作結果を掲載しなさい. (注意: proc.pl0 はなるべく簡単なプログラムにすること.)

課題 6

自分なりの工夫を加えなさい.

提出物

以下のファイルをまとめたアーカイブファイル “自分の学籍番号.tgz”(例 bp13000.tgz).

1. レポートの PDF ファイル report03.pdf
2. コンパイラのソースプログラム (コンパイルに必要なものすべて)
3. コンパイルするための Makefile
4. 拡張した機能を確認するための PL/0’ プログラム
   (a) 配列用 array.pl0
   (b) 手続き用 proc.pl0
   (c) その他の工夫用

アーカイブファイルは, Linux 上の次の操作で作成できる.

1. 自分の学籍番号と同じ名前のディレクトリを作る. たとえば学籍番号が *bp13000* の場合, シェル上で次のような操作をする. (行頭の%はプロンプト)

   % mkdir *bp13000*

2. 作成したディレクトリ内に, 提出するファイルをコピーする.
3. tar コマンドでアーカイブファイルを作成する.

   % tar zcvf *bp13000*.tgz *bp13000*

提出前にアーカイブファイルの内容を確認すること. アーカイブファイルの展開は次のように行う.

% tar zxvf *bp13000*.tgz

提出期限と提出方法

7月 18 日 (土) 中までに、Luminous (https://lmns.sayo.se.shibaura-it.ac.jp) に提出する.

コース名 「言語処理系」

レポート名 「第 3 回レポート課題」

注意点

- コンパイラのコンパイルは, make コマンドでできるようにすること.
- 変更を加えたコンパイラのコンパイルと実行が, 大学の Linux 環境でできるようにすること.