情報工学実験コンパイラ レポート

氏名：佐藤佑太
学生番号：09425566

出題日：平成 27 年 4 月 13 日
提出日：平成 27 年 7 月 28 日
提出期限：平成 27 年 7 月 28 日

1　実験の目的

1. 言語処理系がどのように実装されるのか学ぶ
2. 大きなプログラムにおける関数の切り出しなどの設計・モジュール化
3. コンピュータにとって効率のよい計算方法、アルゴリズムの設計

これらのの三つ事柄について学習することが主な目的である.

2　言語の定義

```
<プログラム> ::= <変数宣言部> <文集合>
<変数宣言部> ::= <宣言文> <変数宣言部> | <宣言文>
<宣言文> ::= define <識別子>;| define <配列宣言>;
<配列宣言>::= <識別子>[<整数>]|<配列宣言>[<整数>]
<文集合> ::=  <文> <文集合>| <文>
<文> ::= <代入文> | <配列代入文> | <ループ文> | <条件分岐文> | <関数宣言文> | <関数>;
<代入文> ::= <識別子> = <算術式>; | <識別子> = call <関数>;
<配列代入文> ::= <配列> = <算術式>; | <配列> = call <関数>;
<算術式> ::= <算術式> <加減演算子> <項> | <項>
<項> ::= <項> <乗除演算子> <因子> | <因子>
<因子> ::= <変数> | (<算術式>)
<加減演算子> ::= + | -
<乗除演算子> ::= * | /
<変数> ::= <識別子> | <数> | <配列>
<引数> ::= <変数>,<引数> | <変数>
<関数> ::= <識別子>(<引数>) |  <識別子>()
<関数宣言文> ::= func <識別子>(<引数>) { <文集合> } | func <識別子>(){ <文集合> }
|func <識別子>(<引数>) {<文集合> return <変数>;}|func <識別子>(){<文集合> return <変数>;}
<ループ文> ::= while (<条件式>) { <文集合> }
<条件分岐文> ::= if (<条件式>) { <文集合> } | if (<条件式>) { <文集合> } else { <文集合> }
<条件式> ::= <算術式> <比較演算子> <算術式>
<比較演算子> ::= == | '<' | '>'| >= | <= | !=
<識別子> ::= <英字> <英数字列> | <英字>
<英数字列> ::= <英数字> <英数字列>| <英数字>
<英数字> ::= <英字> | <数字>
<数> ::= <数><数字> | <0 以外>
<英字> ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w
|x|y|z|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<整数> ::= <数>|0
<数字> ::= 0|1|2|3|4|5|6|7|8|9
<0 以外> ::= 1|2|3|4|5|6|7|8|9
<配列> ::= <識別子>[<変数>]|<配列>[<変数>]
```

3　定義した言語で受理されるプログラムの例

以下が、定義した言語によって受理されるプログラムの例である。

```
1  define array[3];
2  define array2[2][4];
3  define a;
4  define i;
5  define flag;
6
7  func resetArray() {
8    i = 0;
9    while (i<3) {
10     array[i] = 0;
11     i = i + 1;
12   }
```

```
13    return 1;
14  }
15
16  flag = call resetArray();
17  array2[0][flag] = 100;
18
19  if (flag == 1) {
20    array[1] = 1;
21  }else {
22    array[2] = 1;
23  }
```

プログラムは、変数宣言部、関数宣言部、実行部の 3 つに分かれて記述される。

4　字句解析，演算子順位構文解析・再帰下降型構文解析のそれぞれの方法の概略

1. 字句解析プログラムのテキストをある特定の字句に分割し，それぞれに対応したタイプを付与する．(e.g int => 予約語，sum => 識別子， etc..)

2. 演算子順位構文解析算術式を解析するために用いる方法．下向き構文解析法であり，演算子順位行列を用いて構文木を作成する．

   以下に演算子順位行列を示す.

左＼右	+,-	*,/	(	)	$
+,-	>	<	<	>	>
*,/	>	>	<	>	>
(	<	<	<	=	×
)	>	>	×	>	>
$	<	<	<	×	終了

3. 再帰下降型構文解析算術式以外を解析するために用いる方法．言語の定義をそのままプログラムに出来るのでわかりやすい．

5　コード生成の概略

1. メモリの使い方
   最終的に関数を実装することが出来なかったので，すべてのデータはC言語に置けるグローバル変数と扱いは同等となっている．

2. レジスタの使い方

   - $t0 〜 $t6 算術式の解析用
   - $t7 la，li などのデータ読み込み用
   - $t8 条件式の判定用
   - $t9 代入文に置いて配列が代入先になったとき用

3. 算術式のコード生成の方法
   四つ組中間表現を用いている．

4. 特に工夫した点についての説明
   数値または識別子ををを一つだけ代入する場合は，直で$v0 に値が代入されるようになっている．

6　コンパイラのソースプログラムのある場所

http://jikken1.arc.cs.okayama-u.ac.jp/gitbucket/09425566/compiler
実行方法:
　./esp <filename>
<filename>:
　sum.esp, fact.esp

7　最終課題を解くために書いたプログラムの概要

基本的な書き方はC言語に近い．しかし，実装されていない機能が多く存在しているため，書き方に制限がかけられている．データの型はint型しか存在せず，関数は関数名の前にfunとつける必要がある．配列の括弧の中には算術式が入るように再帰的に処理している．

8　最終課題の実行結果

1. 1から10までの数の和
   - プログラム：sum.cm
   - 実行結果：55 (0x10004004)
   - ステップ数：360 instructions
2. 階乗の計算
   - プログラム：fact.cm
   - 実行結果：120 (0x10004004)
   - ステップ数：195 instructions
3. エラトステネスのふるい
   - プログラム：prime.cm
   - 実行結果：2,3,5,7,11...
   - ステップ数：86301 instructions

   ※与えられたアルゴリズムでは探索に無駄が多いため，処理を省いている．
4. 行列積の計算
   - プログラム：matrix.cm
   - 実行結果：19,22,43,50
   - ステップ数：1425 instructions
5. クイックソート
   ローカル変数を作成出来なかったため，作成出来なかった．

9　考察