Федеральное государственное автономное образовательное учреждение высшего образования "Национальный Исследовательский Университет ИТМО" Мегафакультет Компьютерных Технологий и Управления Факультет Программной Инженерии и Компьютерной Техники



Вариант №5
Задание
по дисциплине
'Разработка компиляторов'

Выполнил Студент группы Р33102 Лапин Алексей Александрович Преподаватель: Лаздин Артур Вячеславович

Задание

Разработать язык программирования, который должен реализовать следующие компоненты:

- 1. Присваивание (оператор или операция), арифметические и логические операции.
- 2. Ветвление, включая вариант факультативного else.
- 3. Цикл while.
- 4. Поддержка целочисленного и логического типа данных.
- 5. Многострочные комментарии в стиле Си-подобных языков.

Что добавит баллы:

- 1. Вместо ветвление if [then] else конструируется оператор if elif [elif]+ else
- 2. Вместо цикла while (или в дополнение к нему) конструируется цикл for.

1 Ход работы

Сделал лексический анализатор грамматики с помощью библиотеки FLEX.

```
1 %{
2 #include "ast.h"
3 #include <stdio.h>
4 #include "parser.tab.h"
  %option noyywrap
  %x COMMENT
10
11 %%
13 \mid [0-9] \mid [a-zA-Z] \mid [a-zA-Z0-9] *  { yyerror("Names cannot start with a
      digit");}
                      { yylval.i = atoi(yytext); return NUMBER; }
14 [0-9]+
15
16
       /st tokens for arithmetic and logical operators. st/
17
18
       /* single character ops */
19
  ^{11} \pm ^{11}
20
21
  " * "
  11 / 11
  0 = 0
24
  0,0
25
  0:0
26
27 " ( " |
28 ")" |
29 " { " |
```

```
30 "}"
                 { return yytext[0]; }
31
      /* comparation ops, all are a CMP token */
32
33 ">"
                        { yylval.subtok = NT_GT; return CMP; }
34 " < "
                        { yylval.subtok = NT_LT; return CMP; }
35 "!="
                        { yylval.subtok = NT_NEQ; return CMP; }
36 "=="
                        { yylval.subtok = NT_EQ; return CMP; }
37 " >= "
                        { yylval.subtok = NT_GTE; return CMP; }
38 " <= "
                        { yylval.subtok = NT_LTE; return CMP; }
39
      /* keywords */
40
41 "if"
                        { return IF;
42 "else"
                        { return ELSE;
                                          }
43 "while"
                        { return WHILE;
44 "for"
                        { return FOR;
45 "elif"
                       { return ELIF;
                                          }
46
47 " & & "
                       { return AND;
48 " | "
                       { return OR;
                                         }
49 "and"
                       { return AND;
50 "or"
                       { return OR;
                                         }
51 "not"
                       { return NOT;
52 "!"
                        { return NOT;
                                         }
53
54
     /* booleans */
55
56 "true"
                        { yylval.i = 1; return NUMBER; }
57 "false"
                        { yylval.i = 0; return NUMBER; }
58
       /* comments */
59
60
61 "/*"
                        { BEGIN (COMMENT); }
62 < COMMENT > " */"
                       { BEGIN(INITIAL); }
63 < COMMENT > .
                       { /* consume anything inside comments */ }
64 < COMMENT > \n
                       { /* consume newlines in comments */ }
65 < COMMENT > < < EOF >> { yyerror("Unterminated comment"); }
66
67 "//".*
                      /* single line comment */
68
     /* names */
69
70 | [a-zA-Z] [a-zA-ZO-9] * { yylval.s = strdup(yytext); return NAME; }
71
72
73 [ \t]+
                            { /* ignore whitespace */ }
                            {printf("c> "); } /* ignore line
74 \\\n
    continuation */
                            {
                              }
75 \n
76
                            { printf("Unexpected character: %c\n",
77
     yytext[0]); }
78
79 %%
```

Сделал синтаксический анализатор грамматики с помощью библиотеки BISON.

```
1 %{
2 #include "ast.h"
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <stdarg.h>
6 void yyerror(char *s, ...);
7 int yylex(void);
8 %}
10 %union {
      struct ast *a;
11
      int i;
12
      char *s; /* which symbol */
13
      ntype_t subtok;
14
15 }
16
17 /* declare tokens */
18 %token <i>NUMBER
19 %token <s> NAME
20 %token EOL YYEOF
21 %token IF ELSE WHILE FOR ELIF
22
23 %right '='
24 %left OR
25 %left AND
26 %left NOT
27 %left <subtok > CMP
28 %left '+' '-'
29 %left '*' '/'
30 %nonassoc UMINUS
31 | %type <a> exp list flow if_stmt elif_stmt else_stmt
32
33 %start prog
34 %%
                                            {
35 prog: list YYEOF
                                                   if($1 != NULL) {
36
                                                        print_ast(stdout,
37
                                                           $1, 0);
                                                        FILE* fl =
38
                                                           fopen("out.S",
                                                           "w");
                                                        FILE* f12 =
39
                                                           fopen("tree",
                                                           "w");
                                                        if(fl == NULL) {
40
                                                            printf("Error
41
                                                                opening
                                                                file \n");
                                                            exit(1);
42
                                                        }
43
                                                        if(f12 == NULL) {
44
                                                            printf("Error
45
                                                                opening
```

```
file \n");
                                                            exit(1);
46
                                                        }
47
                                                        print_asm(fl, $1);
48
                                                        print_ast(fl2, $1,
49
                                                           0);
                                                        fclose(fl);
50
                                                        treefree($1);
51
52
                                                   printf("> ");
53
                                               }
       | error YYEOF
                                           { yyerrok; printf("> ");}
55
56
57
58 flow: if_stmt
      | WHILE exp '{' list '}'
                                                            { $$ =
         newflow(NT_WHILE, $2, $4, NULL); }
      | FOR '(' exp ';' exp ';' exp ')' '{' list '}' { $$ =
60
         newfor($3, $5, $7, $10); }
61
      ;
62
63 if_stmt: IF exp '{' list '}' else_stmt
                                                            { $$ =
     newflow(NT_IF, $2, $4, $6); }
     | IF exp '{' list '}' elif_stmt
                                                            { $$ =
          newflow(NT_IF, $2, $4, $6); }
65
66 elif_stmt: ELIF exp '{' list '}' else_stmt
     newflow(NT_IF, $2, $4, $6); }
     | ELIF exp '{' list '}' elif_stmt
                                                            { $$ =
67
         newflow(NT_IF, $2, $4, $6); }
68
69 else_stmt: /* nothing */
                                                            \{ \$\$ = \mathtt{NULL}; \}
      | ELSE '{' list '}'
                                                            \{ \$\$ = \$3; \}
70
71
72 list: /* nothing */
                                               \{ \$\$ = NULL; \}
     | exp ';' list
                                               {
73
                                                   if($3 == NULL) $$ = $1;
74
                                                   else $$ =
75
                                                       newast(NT_LIST, $1,
                                                       $3);
                                               }
76
       | flow list
                                               {
77
                                                   if($2 == NULL) $$ = $1;
78
                                                   else $$ =
79
                                                       newast(NT_LIST, $1,
                                                       $2);
                                               }
80
81
82
                                               \{ \$\$ = newcmp(\$2, \$1, \$3);
83 exp: exp CMP exp
           }
       | exp '+' exp
                                               \{ \$\$ = newast(NT\_ADD, \$1,
          $3);
                    }
```

```
\{ \$\$ = newast(NT_SUB, \$1,
       | exp '-' exp
85
          $3);
                                                  \{ \$\$ = newast(NT_MUL, \$1,
       | exp '*' exp
86
                                                  { $$ = newast(NT_DIV, $1,
       | exp '/' exp
          $3);
       | '(' exp ')'
                                                  \{ \$\$ = \$2;
88
                                                  { $$ = newast(NT_UMINUS, $2,
       | '-' exp
                    %prec UMINUS
89
          NULL);
                     }
                                                  \{ \$\$ = newast(NT\_AND, \$1,
       | exp AND exp
90
          $3);
                   }
                                                  \{ \$\$ = newast(NT_OR, \$1,
       | exp OR exp
91
          $3);
                                                  \{ \$\$ = newast(NT_NOT, \$2,
92
       | NOT exp
          NULL); }
       | NUMBER
                                                  { \$\$ = newnum(\$1);}
93
                          }
       | NAME
                                                  { $$ = newref($1);
94
                          }
                                                  \{ \$\$ = newasgn(\$1, \$3);
       | NAME '=' exp
95
                    }
96
       ;
97 | %%
```

2 Проверка соответсвия требованиям

2.1 Присваивание (оператор или операция)

```
> ./prog
Interactive mode
To execute press Ctrl+D
> x = 4; y = 5; ^D
List
   Assignment x
    Constant 4
   Assignment y
    Constant 5
> %
```

Рис. 1: Присваивание

out.S:

```
jal x1, MAIN
x:
data 0 * 1
```

```
y:
data 0 * 1
MAIN:
li x1, 4
sw x0, 1, x1
li x1, 5
sw x0, 2, x1
ebreak
```

2.2 Арифметические и логические операции. Ветвление, включая вариант факультативного elif.Поддержка целочисленного и логического типа данных.Многострочные комментарии в стиле Си-подобных языков.

Программа:

```
1 \times 4;
2 | y = false;
3 z = 0;
4 if (y) {
      z = 1;
6 }
7 /* Some Comment
8 Multi-line
9 Should be ignored
10 */
|x| = 4
12
      z = 2;
13 }
14 else {
15
      z = 3;
16 }
```

tree:

```
List
Assignment x
    Constant 4
List
    Assignment y
    Constant 0
    List
    Assignment z
        Constant 0
    Ιf
        Symbol y
        Assignment z
        Constant 1
        Ιf
        Comparison ==
            Symbol x
            Constant 4
```

```
Assignment z
Constant 2
Assignment z
Constant 3
```

out.S:

jal x1, MAIN
x:

data 0 * 1

у:

data 0 * 1

7. :

data 0 * 1

MAIN:

li x1, 4

sw x0, 1, x1

li x1, 0

sw x0, 2, x1

li x1, 0

sw x0, 3, x1

lw x2, x0, 2

beq x2, x0, ELSE0

li x1, 1

sw x0, 3, x1

jal x1, ENDIF0

ELSE0:

lw x2, x0, 1

li x3, 4

seq x2, x2, x3

beq x2, x0, ELSE1

li x1, 2

sw x0, 3, x1

jal x1, ENDIF1

ELSE1:

li x1, 3

sw x0, 3, x1

ENDIF1:

ENDIFO:

ebreak

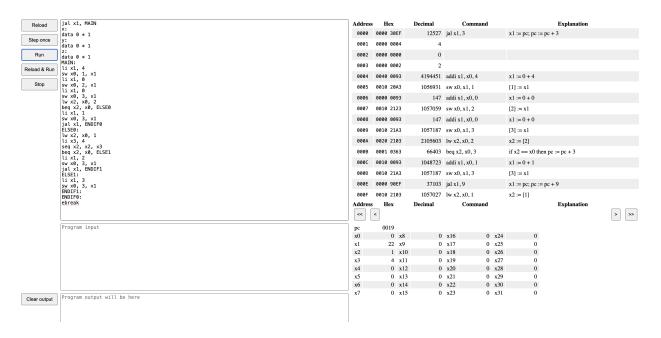


Рис. 2: Программа

2.3 Оператор For

Программа:

tree:

```
List
Assignment x
    Constant 4
List
    Assignment y
    Constant 32
    For
    Assignment i
        Constant 1
    Comparison <
        Symbol i
        Constant 5
    Assignment i
        Operator +
        Symbol i
        Constant 1
    List
        Assignment x
        Operator *
            Symbol x
```

```
Symbol i
        Assignment y
        Operator /
            Symbol y
            Constant 2
out.S:
jal x1, MAIN
i:
data 0 * 1
x:
data 0 * 1
у:
data 0 * 1
MAIN:
li x1, 4
sw x0, 2, x1
li x1, 32
sw x0, 3, x1
li x1, 1
sw x0, 1, x1
WHILEO:
lw x2, x0, 1
li x3, 5
slt x2, x2, x3
beq x2, x0, ENDWHILEO
lw x1, x0, 2
lw x2, x0, 1
mul x1, x1, x2
sw x0, 2, x1
lw x1, x0, 3
li x2, 2
div x1, x1, x2
sw x0, 3, x1
lw x1, x0, 1
li x2, 1
add x1, x1, x2
sw x0, 1, x1
jal x1, WHILEO
ENDWHILEO:
```

ebreak

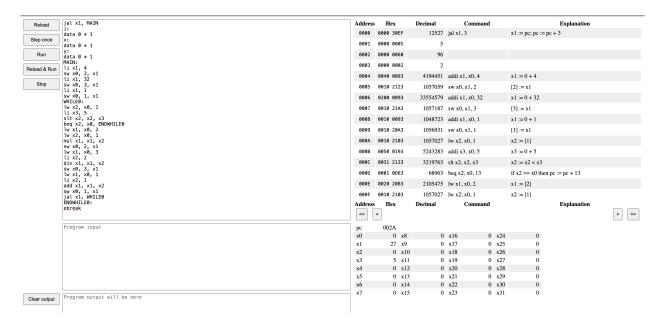


Рис. 3: Программа

2.4 Оператор While

Программа:

```
1  x = 4;
y = 32;
for (i = 1; i < 5; i = i + 1) {
    x = x * i;
    y = y / 2;
}
```

tree:

```
List
  Assignment x
    Constant 2
  While
    Comparison <=
      Symbol x
      Constant 10
    If
      Comparison ==
        Symbol x
        Constant 4
      Assignment x
        Operator +
          Symbol x
          Constant 2
      Assignment x
        Operator +
          Symbol x
          Constant 1
```

```
jal x1, MAIN
x:
data 0 * 1
MAIN:
li x1, 2
sw x0, 1, x1
WHILEO:
lw x2, x0, 1
li x3, 10
seq x4, x2, x3
slt x5, x2, x3
or x2, x4, x5
beq x2, x0, ENDWHILEO
lw x2, x0, 1
li x3, 4
seq x2, x2, x3
beq x2, x0, ELSE0
lw x1, x0, 1
li x2, 2
add x1, x1, x2
sw x0, 1, x1
jal x1, ENDIFO
ELSE0:
lw x1, x0, 1
li x2, 1
add x1, x1, x2
sw x0, 1, x1
ENDIFO:
jal x1, WHILEO
ENDWHILEO:
ebreak
```

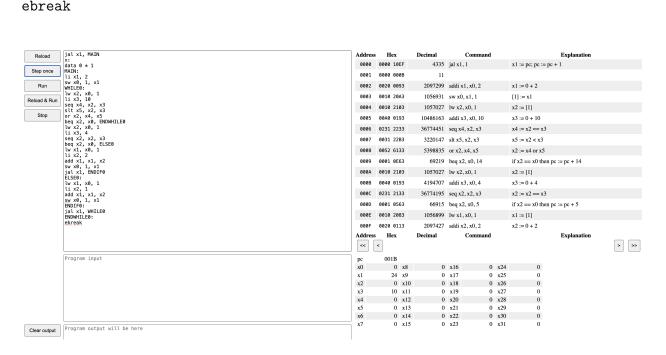


Рис. 4: Программа

3 Обработка ошибок

```
> ./prog
Interactive mode
To execute press Ctrl+D
> 5x = 2;
1: error: Names cannot start with a digit
```

```
> ./prog
Interactive mode
To execute press Ctrl+D
> x = e;
1: error: NameError: name 'e' is not defined
```

```
> ./prog
Interactive mode
To execute press Ctrl+D
> §4
Unexpected character:
```

4 Ссылка на репозиторий

https://github.com/AaLexUser/Compiler-development