

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**

г. Санкт-Петербург  
2024г.

# 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"
5 %}
6
7 %option noyywrap
8
9 %x COMMENT
10
11 %%
12
13 [0-9][a-zA-Z][a-zA-Z0-9]* { yyerror("Names cannot start with a
14 digit");}
15
16 [0-9]+ { yylval.i = atoi(yytext); return NUMBER; }
17
18 /* tokens for arithmetic and logical operators. */
19
20 /* single character ops */
21 "+" |
22 "-" |
23 "*" |
24 "/" |
25 "=" |
26 "," |
27 ";" |
28 "(" |
29 ")" |
30 "{" |
```

```

30 "}"          { return yytext[0]; }
31
32 /* comparation ops, all are a CMP token */
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
40 /* keywords */
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
55 /* booleans */
56 "true"        { yylval.i = 1; return NUMBER; }
57 "false"       { yylval.i = 0; return NUMBER; }
58
59 /* comments */
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
69 /* names */
70 [a-zA-Z][a-zA-Z0-9]* { yylval.s = strdup(yytext); return NAME; }
71
72
73 [ \t]+       { /* ignore whitespace */ }
74 \\n          { printf("c> "); } /* ignore line
continuation */
75 \n          { }
76
77 .           { printf("Unexpected character: %c\n",
yytext[0]); }
78
79 %%

```

Сделал синтаксический анализатор грамматики с помощью библиотеки BISON.



```

46                                     file\n");
47                                     exit(1);
48                                     }
49                                     print_asm(fl, $1);
50                                     print_ast(fl2, $1,
51                                     0);
52                                     fclose(fl);
53                                     treefree($1);
54                                     }
55                                     printf("> ");
56                                     }
57                                     { yyerrok; printf("> ");}
58 flow: if_stmt
59     | WHILE exp '{' list '}' { $$ =
60         newflow(NT_WHILE, $2, $4, NULL); }
61     | FOR '(' exp ';' exp ')' '{' list '}' { $$ =
62         newfor($3, $5, $7, $10); }
63     ;
64 if_stmt: IF exp '{' list '}' else_stmt { $$ =
65     newflow(NT_IF, $2, $4, $6); }
66     | IF exp '{' list '}' elif_stmt { $$ =
67     newflow(NT_IF, $2, $4, $6); }
68     ;
69 elif_stmt: ELIF exp '{' list '}' else_stmt { $$ =
70     newflow(NT_IF, $2, $4, $6); }
71     | ELIF exp '{' list '}' elif_stmt { $$ =
72     newflow(NT_IF, $2, $4, $6); }
73     ;
74 else_stmt: /* nothing */ { $$ = NULL; }
75     | ELSE '{' list '}' { $$ = $3; }
76     ;
77 list: /* nothing */ { $$ = NULL; }
78     | exp ';' list
79     {
80         if($3 == NULL) $$ = $1;
81         else $$ =
82             newast(NT_LIST, $1,
83             $3);
84     }
85     | flow list
86     {
87         if($2 == NULL) $$ = $1;
88         else $$ =
89             newast(NT_LIST, $1,
90             $2);
91     }
92     ;
93 exp: exp CMP exp
94     { $$ = newcmp($2, $1, $3); }
95     | exp '+' exp
96     { $$ = newast(NT_ADD, $1,
97     $3); }

```



```

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 x = 4;
2 y = false;
3 z = 0;
4 if (y) {
5     z = 1;
6 }
7 /* Some Comment
8    Multi-line
9    Should be ignored
10 */
11 elif (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
  If
    Symbol y
    Assignment z
    Constant 1
    If
    Comparison ==
      Symbol x
      Constant 4

```

Assignment z  
Constant 2  
Assignment z  
Constant 3

**out.S:**

```
jal x1, MAIN
x:
data 0 * 1
y:
data 0 * 1
z:
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:
ENDIF0:
ebreak
```



Reload

Step once

Run

Reload & Run

Stop

```

jal x1, MAIN
z:
data 0 * 1
y:
data 0 * 1
z:
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 x2, 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:
ENDIF0:
ebreak

```

Program input

Program output will be here

Clear output

Address	Hex	Decimal	Command	Explanation
0000	0000 30EF	12527	jal x1, 3	x1 := pc; pc := pc + 3
0001	0000 0004	4		
0002	0000 0000	0		
0003	0000 0002	2		
0004	0040 0093	4194451	addi x1, x0, 4	x1 := 0 + 4
0005	0010 20A3	1056931	sw x0, x1, 1	[1] := x1
0006	0000 0093	147	addi x1, x0, 0	x1 := 0 + 0
0007	0010 2123	1057059	sw x0, x1, 2	[2] := x1
0008	0000 0093	147	addi x1, x0, 0	x1 := 0 + 0
0009	0010 21A3	1057187	sw x0, x1, 3	[3] := x1
000A	0020 2103	2105603	lw x2, x0, 2	x2 := [2]
000B	0001 0363	66403	beq x2, x0, 3	if x2 == x0 then pc := pc + 3
000C	0010 0093	1048723	addi x1, x0, 1	x1 := 0 + 1
000D	0010 21A3	1057187	sw x0, x1, 3	[3] := x1
000E	0000 90EF	37103	jal x1, 9	x1 := pc; pc := pc + 9
000F	0010 2103	1057027	lw x2, x0, 1	x2 := [1]

Address

Hex

Decimal

Command

Explanation

<<

>

>>

pc	0019				
x0	0	x8	0	x16	0
x1	22	x9	0	x17	0
x2	1	x10	0	x18	0
x3	4	x11	0	x19	0
x4	0	x12	0	x20	0
x5	0	x13	0	x21	0
x6	0	x14	0	x22	0
x7	0	x15	0	x23	0

Рис. 2: Программа

## 2.3 Оператор For

### Программа:

```

1 x = 4;
2 y = 32;
3 for (i = 1; i < 5; i = i + 1) {
4     x = x * i;
5     y = y / 2;
6 }

```

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
y:
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
WHILE0:
lw x2, x0, 1
li x3, 5
slt x2, x2, x3
beq x2, x0, ENDWHILE0
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, WHILE0
ENDWHILE0:
ebreak
```

Reload

Step once

Run

Reload & Run

Stop

```

jal x1, MAIN
i:
data 0 * 1
x:
data 0 * 1
y:
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
WHILE0:
lw x2, x0, 1
li x3, 5
slt x2, x2, x3
beq x2, x0, ENDWHILE0
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, WHILE0
ENDWHILE0:
ebreak

```

Program input

Program output will be here

Clear output

Address	Hex	Decimal	Command	Explanation
0000	0000 30EF	12527	jal x1, 3	x1 := pc; pc := pc + 3
0001	0000 0005	5		
0002	0000 0060	96		
0003	0000 0002	2		
0004	0040 0093	4194451	addi x1, x0, 4	x1 := 0 + 4
0005	0010 2123	1057059	sw x0, x1, 2	[2] := x1
0006	0200 0093	33554579	addi x1, x0, 32	x1 := 0 + 32
0007	0010 21A3	1057187	sw x0, x1, 3	[3] := x1
0008	0010 0093	1048723	addi x1, x0, 1	x1 := 0 + 1
0009	0010 28A3	1056931	sw x0, x1, 1	[1] := x1
000A	0010 2103	1057027	lw x2, x0, 1	x2 := [1]
000B	0050 0193	5243283	addi x3, x0, 5	x3 := 0 + 5
000C	0031 2133	3219763	slt x2, x2, x3	x2 := x2 < x3
000D	0001 0063	68963	beq x2, x0, 13	if x2 == x0 then pc := pc + 13
000E	0020 2083	2105475	lw x1, x0, 2	x1 := [2]
000F	0010 2103	1057027	lw x2, x0, 1	x2 := [1]

Address

Hex

Decimal

Command

Explanation

<<

<

>

>>

pc	002A				
x0	0	x8	0	x16	0
x1	27	x9	0	x17	0
x2	0	x10	0	x18	0
x3	5	x11	0	x19	0
x4	0	x12	0	x20	0
x5	0	x13	0	x21	0
x6	0	x14	0	x22	0
x7	0	x15	0	x23	0
			0	x30	0
			0	x31	0

Рис. 3: Программа

## 2.4 Оператор While

Программа:

```

1 x = 4;
2 y = 32;
3 for (i = 1; i < 5; i = i + 1) {
4     x = x * i;
5     y = y / 2;
6 }

```

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
WHILE0:
lw x2, x0, 1
li x3, 10
seq x4, x2, x3
slt x5, x2, x3
or x2, x4, x5
beq x2, x0, ENDWHILE0
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, ENDIF0
ELSE0:
lw x1, x0, 1
li x2, 1
add x1, x1, x2
sw x0, 1, x1
ENDIF0:
jal x1, WHILE0
ENDWHILE0:
ebreak

```

Reload
jal x1, MAIN

Step once
x:

Run
data 0 \* 1

Reload & Run
MAIN:

Stop
li x1, 2

sw x0, 1, x1

WHILE0:

lw x2, x0, 1

li x3, 10

seq x4, x2, x3

slt x5, x2, x3

or x2, x4, x5

beq x2, x0, ENDWHILE0

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, ENDIF0

ELSE0:

lw x1, x0, 1

li x2, 1

add x1, x1, x2

sw x0, 1, x1

ENDIF0:

jal x1, WHILE0

ENDWHILE0:

ebreak

Program input

Clear output
Program output will be here

Address	Hex	Decimal	Command	Explanation
0000	0000 10EF	4335	jal x1, 1	x1 := pc; pc := pc + 1
0001	0000 0000	11		
0002	0020 0093	2097299	addi x1, x0, 2	x1 := 0 + 2
0003	0010 20A3	1056931	sw x0, x1, 1	[1] := x1
0004	0010 2103	1057027	lw x2, x0, 1	x2 := [1]
0005	00A0 0193	10486163	addi x3, x0, 10	x3 := 0 + 10
0006	0231 2233	36774451	seq x4, x2, x3	x4 := x2 == x3
0007	0031 22B3	3220147	slt x5, x2, x3	x5 := x2 < x3
0008	0052 6133	5398835	or x2, x4, x5	x2 := x4 or x5
0009	0001 0E63	69219	beq x2, x0, 14	if x2 == x0 then pc := pc + 14
000A	0010 2103	1057027	lw x2, x0, 1	x2 := [1]
000B	0040 0193	4194707	addi x3, x0, 4	x3 := 0 + 4
000C	0231 2133	36774195	seq x2, x2, x3	x2 := x2 == x3
000D	0001 0563	66915	beq x2, x0, 5	if x2 == x0 then pc := pc + 5
000E	0010 2083	1056899	lw x1, x0, 1	x1 := [1]
000F	0020 0113	2097427	addi x2, x0, 2	x2 := 0 + 2

AddressHexDecimalCommandExplanation

<<<>>>

pc	001B
x0	0 x8 0 x16 0 x24 0
x1	24 x9 0 x17 0 x25 0
x2	0 x10 0 x18 0 x26 0
x3	10 x11 0 x19 0 x27 0
x4	0 x12 0 x20 0 x28 0
x5	0 x13 0 x21 0 x29 0
x6	0 x14 0 x22 0 x30 0
x7	0 x15 0 x23 0 x31 0

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>