Студент: А.А.Довженко
Группа: М8О-207Б
Номер по списку: 6

«СИСТЕМЫ ПРОГРАММИРОВАНИЯ»
Курсовой проект 2018.
Часть 1.

Для заданного диалекта языка МИКРОЛИСП на базе класса tCG разработать синтаксически управляемый транслятор (генератор кода) в язык C++.

Работоспособность транслятора проверить на трех контрольных задачах из лабораторных работ №1, №2 и №3:

1. **Определение четности количества единиц в двоичной записи целого неотрицательного числа.**
2. **Решение уравнения методом половинного деления.**
3. **Размен денег.**

Тексты контрольных задач адаптировать к заданному диалекту языка с использованием всех доступных грамматических форм .
Если диалект позволяет сохранить грамматическую форму, примененную в лабораторной работе,

запрещается заменять ее другой формой языка МИКРОЛИСП.

Шаблон файла code-gen.cpp создать с помощью приложения make-code-gen.cpp .

Перечень документов в отчете.

Распечатка грамматики своего варианта задания.

```
>
# $f06
   $id   $idq   $dec   $zero
 $bool   $str     (     )
    +     -     *     /
```

```
    <      =      >     <=
   >=    and    not     or
 cond   else     if    let
define display newline   set!

#
    S -> PROG #1
  PROG -> CALCS1 #2 |
          DEFS #3 |
          DEFS CALCS1 #4
 CALCS1 -> CALCS #5
  CALCS -> CALC #6 |
          CALCS CALC #7
  CALC -> E1 #8 |
          BOOL #9 |
          STR #10 |
          DISPSET #11
    E1 -> E #12
     E -> $id #13 |
          $zero #14 |
          ADD #15 |
          SUB #16 |
          DIV #17 |
          MUL #18 |
          COND #19 |
          CPROC #20
   ADD -> HADD E1 ) #21
  HADD -> ( + #22 |
          HADD E1 #23
   SUB -> HSUB E1 ) #24
  HSUB -> ( - #25 |
          HSUB E1 #26
   DIV -> HDIV E1 ) #27
  HDIV -> ( / #28 |
          HDIV E1 #29
   MUL -> HMUL E1 ) #30
  HMUL -> ( * #31 |
          HMUL E1 #32
  COND -> HCOND CLAUS ) #33
 HCOND -> ( cond #34 |
          HCOND CLAUS #35
 CLAUS -> HCLAUS E1 ) #36
```

```
 HCLAUS -> ( BOOL #37 |
        HCLAUS DISPSET #38
  ELSE -> HELSE E1 ) #39
 HELSE -> ( else #40 |
        HELSE DISPSET #41
   HIF -> ( if BOOL #42
 CPROC -> HCPROC ) #43
HCPROC -> ( $id #44 |
        HCPROC E #45
  BOOL -> $bool #46 |
        CPRED #47 |
        REL #48 |
        OR #49 |
        ( not BOOL ) #50
 CPRED -> HCPRED ) #51
HCPRED -> ( $idq #52 |
        HCPRED E #53
   REL -> HREL E1 ) #54
  HREL -> ( < E #55 |
        ( = E #56
    OR -> HOR BOOL ) #57
   HOR -> ( or #58 |
        HOR BOOL #59
   STR -> $str #60 |
        SIF #61
   SIF -> SIFTRUE STR ) #62
SIFTRUE -> HIF STR #63
   SET -> HSET E1 ) #64
  HSET -> ( set! $id #65
DISPSET -> ( display E1 ) #66 |
        ( display BOOL ) #67 |
        ( display STR ) #68 |
        ( newline ) #69 |
        SET #70
  DEFS -> DEF #71 |
        DEFS DEF #72
   DEF -> PRED #73 |
        VAR #74 |
        PROC #75
  PRED -> HPRED BOOL ) #76
 HPRED -> PDPAR ) #77
 PDPAR -> ( define ( $idq #78 |
```

PDPAR $id #79
VAR -> VARINI) #80
VARINI -> HVAR $zero #81 |
HVAR $dec #82
HVAR -> (define $id #83
PROC -> HPROC LETLOC) #84 |
HPROC E1) #85
HPROC -> PCPAR) #86 |
HPROC DISPSET #87
PCPAR -> (define ($id #88 |
PCPAR $id #89
LETLOC -> HLETLOC E1) #90
HLETLOC -> LTVAR) #91 |
HLETLOC DISPSET #92
LTVAR -> (let (CPROC #93 |
LTVAR CPROC #94

Особенности грамматики по форме GrammaFeatures.rtf . >

1. Вычитание.
1.1* Один и более операндов.
(- x y z)
1.2 Только два операнда.
(- x y)
1.3 Только один операнд.
(- x)
2. Деление.
2.1* Один и более операндов.
(/ x y z)
2.2 Только два операнда.
(/ x y)
2.3 Только один операнд.
(/ x)
3. Числовые литералы токена $zero.
3.1* В общем контексте числового выражения.
0
4. Числовые литералы токена $dec.
4.1 В общем контексте числового выражения.
(+ 1 1)
4.2* Только в определении глобальной переменной.
(define one 1)(+ one one)
4.3 Только в определении процедуры.

(define (one) 1)(+ (one) (one))

5. Форма OR.

5.1* Один и более операндов.

(or #t #f #f)

5.2 Отсутствует.

6. Форма AND.

6.1 Один и более операндов.

(and #t #f #f)

6.2* Отсутствует.

7. Оператор = .

7.1* Есть.

(= x y)

7.2 Отсутствует.

8. Оператор отношения, кроме оператора = .

8.1* (< x y)

8.2 (<= x y)

8.3 (> x y)

8.4 (>= x y)

9. Форма IF для чисел.

9.1 Есть.

(if #t e pi)

9.2* Отсутствует.

10. Форма IF для строк.

10.1* Есть.

(display(if (p?)"Yes""No"))

10.2 Отсутствует.

11. Форма COND.

11.1 Составная ветвь ELSE, ноль и более составных клауз.

(cond(else(display pi)pi))

11.2 Составная ветвь ELSE, одна составная клауза.

(cond((e?)(display e)e)(else(display 0)0))

11.3* Без ветви ELSE, одна и более составных клауз.

(cond(#t(display pi)pi))

11.4 Без ветви ELSE, две составные клаузы.

(cond((e?)(display e)e)(#t(display 0)0))

11.5 Отсутствует.

12. Глобальные переменные.

12.1* Есть

(define a 1)a

12.2 Отсутствуют.

13. Локальные переменные.

13.1* Определяются формой let.
(define (f)(let((a pi))a)) (f)
13.2 Только параметры процедур.
(define(f a) (set! a pi)a) (f 0)

Контрольная задача №1.

Полный протокол трансляции без трассировки (крупный белый шрифт на ярком черном фоне).
>

```
karma@DESKTOP-K0CDBM7:/mnt/c/Users/Karma/Desktop/sp18/curs1new$ g++ mlispgen.cpp
karma@DESKTOP-K0CDBM7:/mnt/c/Users/Karma/Desktop/sp18/curs1new$ ./a.out
Input gramma name>f06
Gramma:f06.txt
Source>even-odd
Source:even-odd.ss
   1|(define one 1)
   2|(define two 2)
   3|(define million 1000000)
   4|(define ten-thousand 10000)
   5|
   6|(define(even-bits n)
   7|  (cond((= n 0)one)
   8|        ((=(remainder n two)0)
   9|           (even-bits (quotient n two)))
  10|        (#t(odd-bits(quotient n two)))
  11|  )
  12|)
  13|
  14|(define(odd-bits n)
  15|  (cond((= n 0)0)
  16|        ((=(remainder n two)0)
  17|           (odd-bits (quotient n two)))
  18|        (#t(even-bits(quotient n two)))
  19|        ))
  20|(define(display-bin n)
  21|  (display(remainder n two))
  22|  (cond((= n 0)0)
  23|        (#t (display-bin (quotient n two)))
  24|  )
  25|)
  26|
  27|
  28|(define(report-results n)
  29|  (display "Happy birthday to you!\n\t")
  30|  (display n)(display " (decimal)\n\t")
  31|  (display "\teven?\t")(display (if(=(even-bits n)one) "yes" "no"))
  32|  (newline)
  33|  (display "\todd?\t")(display (if(=(odd-bits n)one) "yes" "no"))
  34|  (newline)
  35|  (set! n(display-bin n))(display "(reversed binary)\n")
```

```
  0
       )
;***** Date of YOUR birthday *******
(define dd 2)
(define mm 12)
(define yyyy 1997)
;**********************************
(report-results (+ (* dd million)
                   (* mm ten-thousand)
                   yyyy))

```

Code:

```
/*  AAD2018   */
#include "mlisp.h"
extern double one;
extern double two;
extern double million;
extern double ten__thousand;
double even__bits(double n);
double odd__bits(double n);
double display__bin(double n);
double report__results(double n);
extern double dd;
extern double mm;
extern double yyyy;
//________________
double one = 1;
double two = 2;
double million = 1000000;
double ten__thousand = 10000;
double even__bits(double n){
        return ((n == 0) ? (one) :
                (remainder(n, two) == 0) ? (even__bits(quotient(n, two))) :
                 true ? (odd__bits(quotient(n, two))) :
                 _infinity);
}
double odd__bits(double n){
        return ((n == 0) ? (0) :
                (remainder(n, two) == 0) ? (odd__bits(quotient(n, two))) :
                 true ? (even__bits(quotient(n, two))) :
```

```
                _infinity);
}
double display__bin(double n){
        display(remainder(n, two));
        return ((n == 0) ? (0) :
                true ? (display__bin(quotient(n, two))) :
                _infinity);
}
double report__results(double n){
        display("Happy birthday to you!\n\t");
        display(n);
        display(" (decimal)\n\t");
        display("\teven?\t");
        display((even__bits(n) == one) ? "yes" : "no");
        newline();
        display("\todd?\t");
        display((odd__bits(n) == one) ? "yes" : "no");
        newline();
         n = display__bin(n);
        display("(reversed binary)\n");
        return 0;
}
double dd = 2;
double mm = 12;
double yyyy = 1997;
int main(void){
 display(report__results((dd * million + mm * ten__thousand + yyyy))); newline();
 std::cin.get();
 return 0;
}
```

Протокол запуска задачи на C++.

>

```
karma@DESKTOP-K0CDBM7:/mnt/c/Users/Karma/Desktop/sp18/curs1new$ g++ even-odd.cpp
karma@DESKTOP-K0CDBM7:/mnt/c/Users/Karma/Desktop/sp18/curs1new$ ./a.out
Happy birthday to you!
        2121997 (decimal)
        even?   no
        odd?    yes
10110000100001100000010(reversed binary)
0
```

Протокол запуска задачи на Лиспе.

>

```
Добро пожаловать в DrRacket, версия 5.3.6 [3m].
Язык: Pretty Big; memory limit: 128 MB.
Happy birthday to you!
        2121997 (decimal)
        even?   no
        odd?    yes
10110000100001100000010(reversed binary)
0
>
```

Контрольная задача №2.

Полный протокол трансляции без трассировки (крупный белый шрифт на ярком черном фоне).

>

```
Source>half-interval
Source:half-interval.ss
  1|;  half-interval.ss 2018
  2|(define one 1)
  3|(define two 2)
  4|(define three 3)
  5|(define one-hundred-six 106)
  6|(define one-hundred-seven 107)
  7|(define zero-point-twenty-five 0.25)
  8|(define one-point-two-thousand-and-two 1.2502)
  9|(define tolerance 1.0e-5)
 10|
 11|(define (half-interval-metod a b)
 12|  (let((a-value (fun a))
 13|       (b-value (fun b))
 14|      )
 15|      (cond((not(or(not(< a-value 0))(not(< 0 b-value))))
 16|                (try a b))
 17|          ((not(or(not(< 0 a-value))(not(< b-value 0))))
 18|                (try b a))
 19|         (#t(+ b one))
 20|     )
 21|  )
 22|)
 23|(define(try neg-point pos-point)
 24| (let(
 25|       (midpoint (average neg-point pos-point))
 26|       (test-value 0)
 27|     )
 28|     (display "+")
 29|     (cond((close-enough? neg-point pos-point) midpoint)
 30|        (#t (set! test-value (fun midpoint))
 31|            (cond((< 0 test-value)(try neg-point midpoint))
 32|                 ((< test-value 0)(try midpoint pos-point))
 33|                 (#t midpoint))
 34|         )
 35|     )
```

```
 36| )
 37|)
 38|(define (close-enough? x y)
 39|  (<(abs (- x y))tolerance))
 40|(define (average x y)(/(+ x y) two))
 41|(define (root a b)
 42| (display"interval=\t[")
 43| (display a)
 44| (display" , ")
 45| (display b)
 46| (display"]\n")
 47| (let((temp (half-interval-metod a b)))
 48|      (newline)
 49|      (display"discrepancy=\t")
 50|      (display(fun temp))(newline)
 51|      (display"root=\t\t")
 52|      (display(if(=(- temp b one)0)"[bad]" "[good]"))
 53|      temp
 54| )
 55|)
 56|
 57|(define(fun z)
 58|  (set! z (- z (/ one-hundred-six one-hundred-seven)(/ e)))
 59|  (+ (* zero-point-twenty-five (expt z three))
 60|   (- z one-point-two-thousand-and-two)
 61|  )
 62|)
 63|
 64|
 65|" AAD variant 6"
 66|(root two three)
 67|
```

Code:

```
/*  AAD2018   */
#include "mlisp.h"
extern double one;
extern double two;
extern double three;
extern double one__hundred__six;
extern double one__hundred__seven;
```

```
extern double zero__point__twenty__five;
extern double one__point__two__thousand__and__two;
extern double tolerance;
double half__interval__metod(double a, double b);
double __AAD2018__try(double neg__point, double pos__point);
bool close__enough_Q(double x, double y);
double average(double x, double y);
double root(double a, double b);
double fun(double z);
//________________
double one = 1;
double two = 2;
double three = 3;
double one__hundred__six = 106;
double one__hundred__seven = 107;
double zero__point__twenty__five = 0.25;
double one__point__two__thousand__and__two = 1.2502;
double tolerance = 1.0e-5;
double half__interval__metod(double a, double b){
        {//let
        double a__value(fun(a)),
        b__value(fun(b));
        return ((!((!(a__value < 0)) || (!(0 < b__value)))) ? (__AAD2018__try(a, b)) :
               (!((!(0 < a__value)) || (!(b__value < 0)))) ? (__AAD2018__try(b, a)) :
                true ? ((b + one)) :
                _infinity);
        }//let
}
double __AAD2018__try(double neg__point, double pos__point){
        {//let
        double midpoint(average(neg__point, pos__point)),
        test__value(0);
        display("+");
        return (close__enough_Q(neg__point, pos__point) ? (midpoint) :
                true ?  test__value = fun(midpoint),
(((0 < test__value) ? (__AAD2018__try(neg__point, midpoint)) :
               (test__value < 0) ? (__AAD2018__try(midpoint, pos__point)) :
                true ? (midpoint) :
                _infinity)) :
                _infinity);
        }//let
```

```
}
bool close__enough_Q(double x, double y){
        return (abs((x - y)) < tolerance);
}
double average(double x, double y){
        return (x + y) / two;
}
double root(double a, double b){
        display("interval=\t[");
        display(a);
        display(" , ");
        display(b);
        display("]\n");
        {//let
        double temp(half__interval__metod(a, b));
        newline();
        display("discrepancy=\t");
        display(fun(temp));
        newline();
        display("root=\t\t");
        display(((temp - b - one) == 0) ? "[bad]" : "[good]");
        return temp;
        }//let
}
double fun(double z){
         z = (z - one__hundred__six / one__hundred__seven - 1 / e);
        return (zero__point__twenty__five * expt(z, three) + (z - one__point__two__thousand__and__two));
}
int main(void){
 display(" AAD variant 6"); newline();
 display(root(two, three)); newline();
 std::cin.get();
 return 0;
}
```

Протокол запуска задачи на C++.

>

```
karma@DESKTOP-K0CDBM7:/mnt/c/Users/Karma/Desktop/sp18/curs1new$ g++ half-interval.cpp
karma@DESKTOP-K0CDBM7:/mnt/c/Users/Karma/Desktop/sp18/curs1new$ ./a.out
 AAD variant 6
interval=       [2 , 3]
++++++++++++++++++
discrepancy=    -2.684890032134124e-06
root=           [good]2.358646392822266
```

Протокол запуска задачи на Лиспе.

>

```
Добро пожаловать в DrRacket, версия 5.3.6 [3m].
Язык: Pretty Big; memory limit: 128 MB.
" AAD variant 6"
interval=       [2 , 3]
++++++++++++++++++
discrepancy=    -2.68489003252270 2e-006
root=           [good]#e2.35864639282226 5625
```

Контрольная задача №3.

Полный протокол трансляции без трассировки (крупный белый шрифт на ярком черном фоне).

>

```
Source>coin
Source:coin.ss
   1|(define dd 2)
   2|(define mm 12)
   3|(define LAGEST-COIN 10)
   4|
   5|(define one 1)
   6|(define two 2)
   7|(define three 3)
   8|(define five 5)
   9|(define ten 10)
  10|(define hundred 100)
  11|
  12|(define (cc amount largest-coin)
  13|    (cond((or (= amount 0)(= largest-coin one))
  14|        one)
  15|        ((not (not(or (not(< 0 amount)) (not(< 0 largest-coin)))))
  16|        0)
  17|        (#t (+ (cc amount (next-coin largest-coin)) (cc (- amount largest-coin) largest-coin)
  18|        ))
  19|    )
  20|)
  21|
  22|(define (count-change amount)
  23|    (cc amount LAGEST-COIN)
  24|)
  25|
  26|(define (next-coin coin)
  27|    (cond((= coin ten) five)
  28|        ((= coin five) three)
  29|        ((= coin three) two)
  30|        ((= coin two) one)
  31|        (#t 0)
  32|    )
  33|)
  34|
  35|(define (GR-AMOUNT) (+ (* hundred mm) dd))
  36|
```

```
 37|(display " AAD variant 6")(newline)
 38|(display " 1-2-3-5-10")(newline)
 39|(display "count__change for 100 \t= ")
 40|(display (count-change hundred))(newline)
 41|(display "count__change for ")
 42|(display (GR-AMOUNT))
 43|(display " \t= ")
 44|(display(count-change (GR-AMOUNT)))(newline)
 45|
```

_____

Code:

```
/*  AAD2018   */
#include "mlisp.h"
extern double dd;
extern double mm;
extern double LAGEST__COIN;
extern double one;
extern double two;
extern double three;
extern double five;
extern double ten;
extern double hundred;
double cc(double amount, double largest__coin);
double count__change(double amount);
double next__coin(double coin);
double GR__AMOUNT();
//_____________
double dd = 2;
double mm = 12;
double LAGEST__COIN = 10;
double one = 1;
double two = 2;
double three = 3;
double five = 5;
double ten = 10;
double hundred = 100;
double cc(double amount, double largest__coin){
       return (((amount == 0) || (largest__coin == one)) ? (one) :
              (!(!((!(0 < amount)) || (!(0 < largest__coin))))) ? (0) :
               true ? ((cc(amount, next__coin(largest__coin)) + cc((amount - largest__coin), largest__coin))) :
               _infinity);
```

```
}
double count__change(double amount){
        return cc(amount, LAGEST__COIN);
}
double next__coin(double coin){
        return ((coin == ten) ? (five) :
                (coin == five) ? (three) :
                (coin == three) ? (two) :
                (coin == two) ? (one) :
                 true ? (0) :
                 _infinity);
}
double GR__AMOUNT(){
        return (hundred * mm + dd);
}
int main(void){
display(" AAD variant 6");
newline();
display(" 1-2-3-5-10");
newline();
display("count__change for 100 \t= ");
display(count__change(hundred));
newline();
display("count__change for ");
display(GR__AMOUNT());
display(" \t= ");
display(count__change(GR__AMOUNT()));
newline();
 std::cin.get();
 return 0;
}
```

Протокол запуска задачи на C++.

>

```
karma@DESKTOP-K0CDBM7:/mnt/c/Users/Karma/Desktop/sp18/curs1new$ g++ coin.cpp
karma@DESKTOP-K0CDBM7:/mnt/c/Users/Karma/Desktop/sp18/curs1new$ ./a.out
 AAD variant 6
 1-2-3-5-10
count__change for 100   = 20592
count__change for 1202  = 300174622
```

Протокол запуска задачи на Лиспе.

>

```
Добро пожаловать в DrRacket, версия 5.3.6 [3m].
Язык: Pretty Big; memory limit: 128 MB.
 AAD variant 6
 1-2-3-5-10
count__change for 100   = 20592
count__change for 1202  = 300174622
```

Распечатка файла code-gen.cpp.
>

```
/* $f06 */
#include "code-gen.h"
using namespace std;

int tCG::p01(){ // S -> PROG
  string header ="/*  " + lex.Authentication()+"   */\n";
  header += "#include \"mlisp.h\"\n";
  header += declarations;
  header += "//____________________ \n";
  S1->obj = header + S1->obj;
    return 0;
}

int tCG::p02() { //   PROG -> CALCS1
    S1->obj = "int main(){\n" + S1->obj
    + " std::cin.get();\n return 0;\n}\n";
    return 0;
}

int tCG::p03() { //   PROG -> DEFS
    S1->obj += "int main(void){\n"
    " display(\"No calculations!\");newline();\n"
    " std::cin.get();\nreturn 0;\n}\n";
    return 0;
}

int tCG::p04() { //   PROG -> DEFS CALCS1
    S1->obj += "int main(void){\n" + S2->obj + "
std::cin.get();\n return 0;\n}\n";
    return 0;
}

int tCG::p05() { // CALCS1 -> CALCS
    return 0;
}

int tCG::p06() { //  CALCS -> CALC
    return 0;
}
```

```
int tCG::p07() { //  CALCS -> CALCS CALC
   S1->obj += S2->obj;
   return 0;
}

int tCG::p08() { //   CALC -> E1
   S1->obj = " display(" + S1->obj + "); newline();\n";
   return 0;
}

int tCG::p09() { //   CALC -> BOOL
   S1->obj = " display(" + S1->obj + "); newline();\n";
   return 0;
}

int tCG::p10() { //   CALC -> STR
   S1->obj = " display(" + S1->obj + "); newline();\n";
   return 0;
}

int tCG::p11() { //   CALC -> DISPSET
   S1->obj += ";\n";
   return 0;
}

int tCG::p12() { //    E1 -> E
   return 0;
}

int tCG::p13() { //     E -> $id
   S1->obj = decor(S1->name);
   return 0;
}

int tCG::p14() { //     E -> $zero
   S1->obj = S1->name;
   return 0;
}

int tCG::p15() { //     E -> ADD
   return 0;
}
```

```
int tCG::p16() { //     E -> SUB
   return 0;
}

int tCG::p17() { //     E -> DIV
   return 0;
}

int tCG::p18() { //     E -> MUL
   return 0;
}

int tCG::p19() { //     E -> COND
   return 0;
}

int tCG::p20() { //     E -> CPROC
   return 0;
}

int tCG::p21() { //    ADD -> HADD E1 )
   if (S1->count == 0) {
      S1->obj = S2->obj;
   } else {
      S1->obj += S2->obj + ")";
   }
   S1->count = 0;
   return 0;
}

int tCG::p22() { //   HADD -> ( +
   S1->obj = "(";
   return 0;
}

int tCG::p23() { //   HADD -> HADD E1
   S1->obj += S2->obj + " + ";
   ++S1->count;
   return 0;
}
```

```
int tCG::p24() { //    SUB -> HSUB E1 )
   if (S1->count == 1) {
      S1->obj += "-" + S2->obj + ")";
   } else {
      S1->obj += S2->obj + ")";
   }
   S1->count = 0;
   return 0;
}

int tCG::p25() { //   HSUB -> ( -
   S1->obj = "(";
   S1->count = 1;
   return 0;
}

int tCG::p26() { //   HSUB -> HSUB E1
   S1->obj += S2->obj + " - ";
   ++S1->count;
   return 0;
}

int tCG::p27() { //    DIV -> HDIV E1 )
   if (S1->count == 0) {
      S1->obj = "1 / " + S2->obj;
   } else {
      S1->obj += S2->obj;
   }
   S1->count = 0;
   return 0;
}

int tCG::p28() { //   HDIV -> ( /
   return 0;
}

int tCG::p29() { //   HDIV -> HDIV E1
   S1->obj += S2->obj + " / ";
   ++S1->count;
   return 0;
}
```

```
int tCG::p30() { //    MUL -> HMUL E1 )
   if (S1->count == 0) {
      S1->obj = S2->obj;
   } else {
      S1->obj += S2->obj;
   }
   S1->count = 0;
   return 0;
}

int tCG::p31() { //   HMUL -> ( *
   return 0;
}

int tCG::p32() { //   HMUL -> HMUL E1
   S1->obj += S2->obj + " * ";
   ++S1->count;
   return 0;
}

int tCG::p33() { //   COND -> HCOND CLAUS )
   S1->obj += " " + S2->obj + " _infinity)";
   return 0;
}

int tCG::p34() { //  HCOND -> ( cond
   S1->obj = "(";
   return 0;
}

int tCG::p35() { //  HCOND -> HCOND CLAUS
   S1->obj += S2->obj;
   S1->count = 0;
   return 0;
}

int tCG::p36() { //  CLAUS -> HCLAUS E1 )
   S1->obj += "(" + S2->obj + ") : \n\t\t";
   return 0;
}

int tCG::p37() { // HCLAUS -> ( BOOL
```

```
   S1->obj += S2->obj + " ? ";
   return 0;
}

int tCG::p38() { // HCLAUS -> HCLAUS DISPSET
   S1->obj += S2->obj + ",\n";
   return 0;
}

int tCG::p39() { //   ELSE -> HELSE E1 )
   return 0;
}

int tCG::p40() { //  HELSE -> ( else
   return 0;
}

int tCG::p41() { //  HELSE -> HELSE DISPSET
   return 0;
}

int tCG::p42() { //    HIF -> ( if BOOL
   S1->obj += S3->obj + " ? ";
   return 0;
}

int tCG::p43() { //  CPROC -> HCPROC )
   if (!S1->count) {
      S1->obj += "(";
   }
   S1->obj += ")";
   return 0;
}

int tCG::p44() { // HCPROC -> ( $id
   S1->obj += decor(S2->name);
   S1->count = 0;
   return 0;
}

int tCG::p45() { // HCPROC -> HCPROC E
   if (!S1->count) {
```

```
      S1->obj += "(";
   } else {
      S1->obj += ", ";
   }
   S1->obj += S2->obj;
   ++S1->count;
   return 0;
}

int tCG::p46() { //   BOOL -> $bool
   S1->obj = (S1->name == "#t" ? "true" : "false");
   return 0;
}

int tCG::p47() { //   BOOL -> CPRED
   return 0;
}

int tCG::p48() { //   BOOL -> REL
   return 0;
}

int tCG::p49() { //   BOOL -> OR
   return 0;
}

int tCG::p50() { //   BOOL -> ( not BOOL )
   S1->obj += "(!" + S3->obj + ")";
   return 0;
}

int tCG::p51() { //  CPRED -> HCPRED )
   S1->obj += ")";
   return 0;
}

int tCG::p52() { // HCPRED -> ( $idq
   S1->obj += decor(S2->name);
   return 0;
}

int tCG::p53() { // HCPRED -> HCPRED E
```

```
   if (!S1->count) {
      S1->obj += "(";
   } else {
      S1->obj += ", ";
   }
   S1->obj += S2->obj;
   ++S1->count;
   return 0;
}

int tCG::p54() { //    REL -> HREL E1 )
   S1->obj += S2->obj + S3->name;
   return 0;
}

int tCG::p55() { //   HREL -> ( < E
   S1->obj += "(" + S3->obj + " < ";
   return 0;
}

int tCG::p56() { //   HREL -> ( = E
   S1->obj += S1->name + S3->obj + " " + S2->name + S2-
>name + " ";
   return 0;
}

int tCG::p57() { //     OR -> HOR BOOL )
   if (!S1->count) {
      S1->obj = S2->obj;
   } else {
      S1->obj += S2->obj + ")";
   }
   S1->count = 0;
   return 0;
}

int tCG::p58() { //    HOR -> ( or
   S1->obj = "(";
   return 0;
}

int tCG::p59() { //    HOR -> HOR BOOL
```

```
   S1->obj += S2->obj + " || ";
   ++S1->count;
   return 0;
}

int tCG::p60() { //    STR -> $str
   S1->obj = S1->name;
   return 0;
}

int tCG::p61() { //    STR -> SIF
   return 0;
}

int tCG::p62() { //    SIF -> SIFTRUE STR )
   S1->obj += " : " + S2->obj;
   return 0;
}

int tCG::p63() { //SIFTRUE -> HIF STR
   S1->obj += S2->obj;
   return 0;
}

int tCG::p64() { //    SET -> HSET E1 )
   if (S1->count) {
      S1->obj += ", ";
   }
   S1->obj += S2->obj;
   return 0;
}

int tCG::p65() { //   HSET -> ( set! $id
   S1->obj = " " + decor(S3->name) + " = ";
   return 0;
}

int tCG::p66() { //DISPSET -> ( display E1 )
   S1->obj = "display(" + S3->obj + ")";
   return 0;
}
```

```
int tCG::p67() { //DISPSET -> ( display BOOL )
   S1->obj = "display(" + S3->obj + ")";
   return 0;
}

int tCG::p68() { //DISPSET -> ( display STR )
   S1->obj = "display(" + S3->obj + ")";
   return 0;
}

int tCG::p69() { //DISPSET -> ( newline )
   S1->obj = "newline()";
   return 0;
}

int tCG::p70() { // DISPSET -> SET
   return 0;
}

int tCG::p71() { //   DEFS -> DEF
   return 0;
}

int tCG::p72() { //   DEFS -> DEFS DEF
   S1->obj += S2->obj;
   return 0;
}

int tCG::p73() { //    DEF -> PRED
   return 0;
}

int tCG::p74() { //    DEF -> VAR
   S1->obj += ";\n";
   declarations += "extern " + decor(S1->name) + ";\n";
   return 0;
}

int tCG::p75() { //    DEF -> PROC
   return 0;
}
```

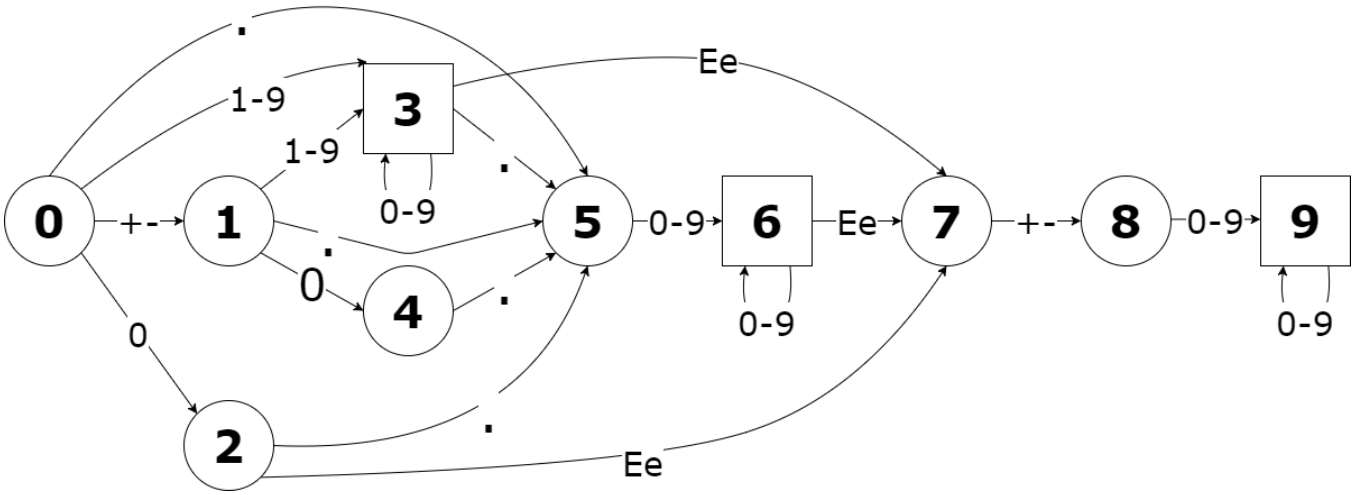
```
int tCG::p76() { //   PRED -> HPRED BOOL )
   S1->obj += S2->obj + ";\n}\n";
   return 0;
}

int tCG::p77() { //  HPRED -> PDPAR )
   S1->obj += ")";
   declarations += S1->obj + ";\n";
   S1->obj += "{\n\treturn ";
   return 0;
}

int tCG::p78() { //  PDPAR -> ( define ( $idq
   S1->obj = "bool " + decor(S4->name) + "(";
   S1->count = 0;
   return 0;
}

int tCG::p79() { //  PDPAR -> PDPAR $id
   if (S1->count) {
      S1->obj += ", ";
   }
   S1->obj += "double " + decor(S2->name);
   ++S1->count;
   return 0;
}

int tCG::p80() { //    VAR -> VARINI )
   return 0;
}

int tCG::p81() { // VARINI -> HVAR $zero
   S1->name = S1->obj;
   S1->obj += " = " + S2->name;
   return 0;
}

int tCG::p82() { // VARINI -> HVAR $dec
   S1->name = S1->obj;
   S1->obj += " = " + S2->name;
   return 0;
}
```

```
int tCG::p83() { //   HVAR -> ( define $id
   S1->obj = "double " + decor(S3->name);
   return 0;
}

int tCG::p84() { //   PROC -> HPROC LETLOC )
   S1->obj += S2->obj + "}\n";
   return 0;
}

// + конец процедуры (ретурн)
int tCG::p85() { //   PROC -> HPROC E1 )
   S1->obj += "\treturn " + S2->obj + ";\n}\n";
   return 0;
}

int tCG::p86() { //  HPROC -> PCPAR )
   S1->obj += ")";
   declarations += S1->obj + ";\n";
   S1->obj += "{\n";
   S1->count = 0;
   return 0;
}

int tCG::p87() { //  HPROC -> HPROC DISPSET
   S1->obj += "\t" + S2->obj + ";\n";
   return 0;
}

int tCG::p88() { //  PCPAR -> ( define ( $id
   S1->obj = "double " + decor(S4->name) + "(";
   S1->count = 0;
   return 0;
}

int tCG::p89() { //  PCPAR -> PCPAR $id
   if (S1->count) {
      S1->obj += ", ";
   }
   S1->obj += "double " + decor(S2->name);
   ++S1->count;
```

```
   return 0;
}

int tCG::p90() { // LETLOC -> HLETLOC E1 )
   S1->obj +=  "\treturn " + S2->obj +";\n" +
"\t}//let\n";
   return 0;
}

int tCG::p91() { //HLETLOC -> LTVAR )
   S1->obj += ";\n";
   return 0;
}

int tCG::p92() { //HLETLOC -> HLETLOC DISPSET
   S1->obj += "\t" + S2->obj + ";\n";
   return 0;
}

int tCG::p93() { //  LTVAR -> ( let ( CPROC
   S1->obj += "\t{//let\n\tdouble " + S4->obj;
   return 0;
}

int tCG::p94() { //  LTVAR -> LTVAR CPROC
   S1->obj += ",\n\t" + S2->obj;
      return 0;
}
//_____________________________
int tCG::p95(){return 0;} int tCG::p96(){return 0;}
int tCG::p97(){return 0;} int tCG::p98(){return 0;}
int tCG::p99(){return 0;} int tCG::p100(){return 0;}
int tCG::p101(){return 0;} int tCG::p102(){return 0;}
int tCG::p103(){return 0;} int tCG::p104(){return 0;}
int tCG::p105(){return 0;} int tCG::p106(){return 0;}
int tCG::p107(){return 0;} int tCG::p108(){return 0;}
int tCG::p109(){return 0;} int tCG::p110(){return 0;}
```

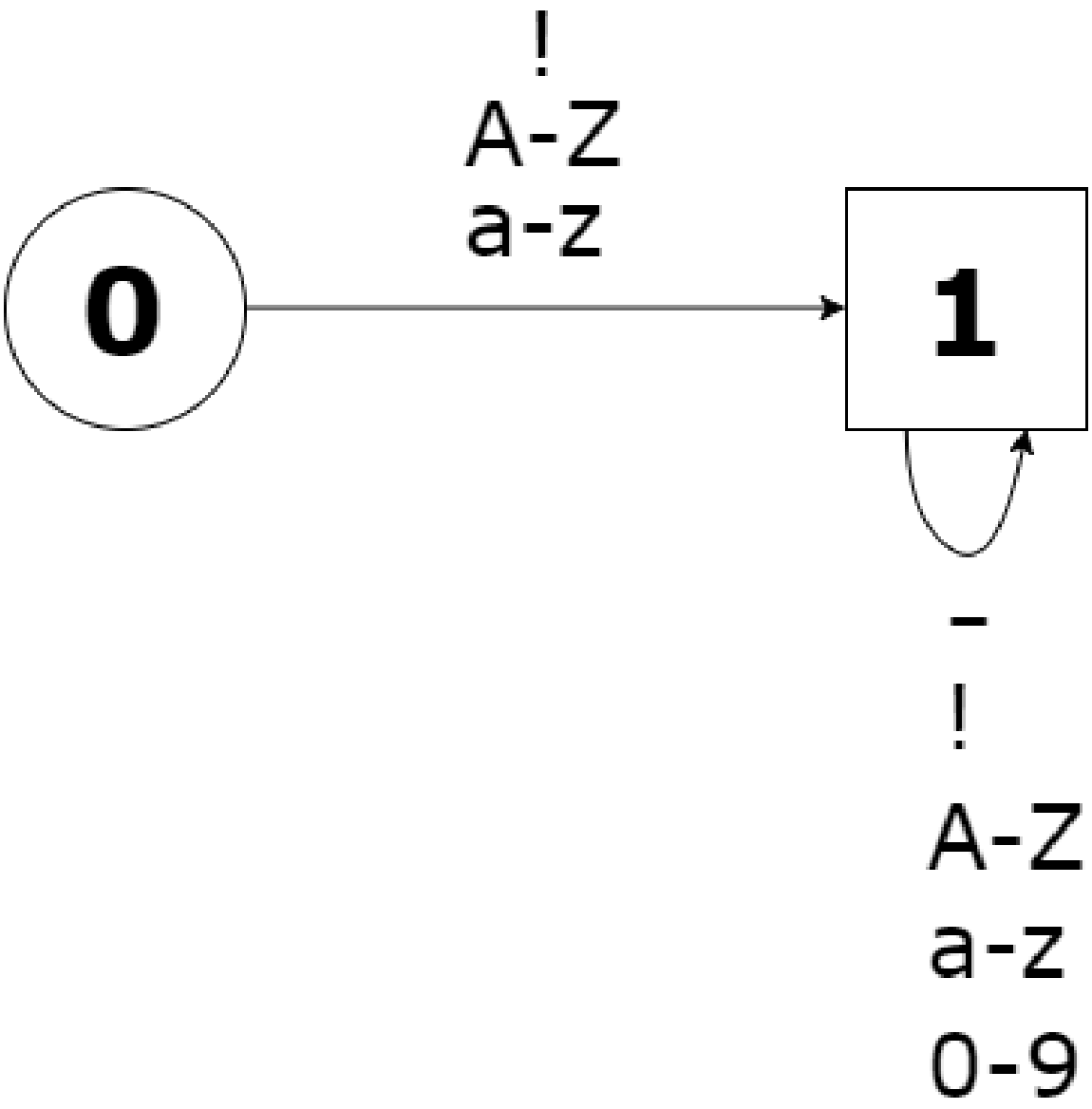
Диаграммы автоматов из лабораторной работы №5 для токенов $dec, $id, $idq. Над каждой диаграммой проставить номер варианта шаблона токена и его краткое описание. Все диаграммы должны быть построены в одном редакторе и должны иметь единый стиль изображения.
>
$dec: 2. Целую часть можно опустить, СОХРАНЯЯ точку и дробную часть, например, .5, -.5, +.5, .5e+0 .

$dec:



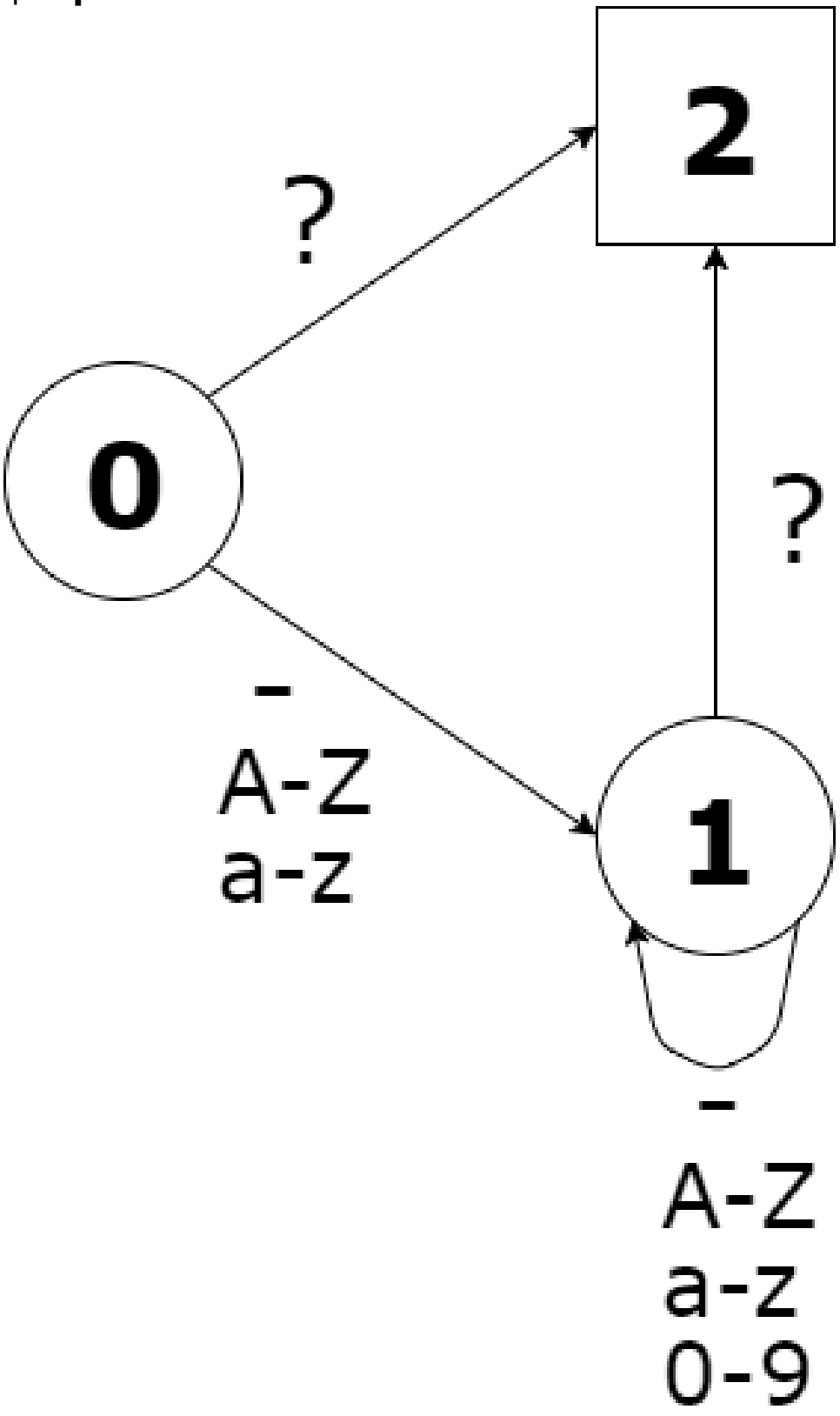
$id: 6. Дополнительных ограничений нет.

$id:



$idq: 1. ‘?’ можно использовать только один раз, и только на последнем месте.

$idq:



Выводы по проделанной работе - не менее одной страницы не разбавленного «водой» текста.
>

Построение синтаксического транслятора стало для меня сложной, но выполнимой задачей. Основные трудности возникли из-за того, что в начале выполнения работы я пыталась сделать все и сразу. Такой подход не принес ничего хорошего – логика разных транслирующих процедур постоянно пересекалась и было сложно отследить, какая из них работает неправильно. Но когда я поняла, что лучше изолировать отдельные языковые конструкции, построить для них дерево разбора, на основе которого написать процедуры трансляции по принципу «от меньшего к большему», задача сильно упростилась. Потом оставалось собирать их как пазл. Самым сложным конкретно в моей грамматике стало определение локальных переменных.

По завершению у меня получился рабочий транслятор. При запуске программ на разных языках их выводы эквиваленты, что не может не радовать. Значит все верно.

Стоит добавить, что теория построения компиляторов шагнула далеко вперед, и теперь, чтобы сделать парсер или лексер такого сложного языка как например C++, достаточно просто описать его грамматику и запустить LEXX/Bison. Эти утилиты сами сгенерируют код парсера и лексера – нам ничего делать не надо. Теория языков разработана почти полностью, а главное – полностью автоматизирована. Лексеры и парсеры необходимо знать на каком-то базовом уровне скорее для общего развития (часто возникают похожие задачи, не связанные напрямую с языками, и эти знания помогают).

Задание первой части курсового проекта выполнено в полном объеме.