**Студент: Дубинин А. О.**
**Группа: М8О-206-17**
**Номер по списку: 7**

**«СИСТЕМЫ ПРОГРАММИРОВАНИЯ»**
**Курсовая работа 2019.**
**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. **Поиск минимума функции методом золотого сечения.**
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.**

```
># $a07
   $id   $idq   $dec   $zero
 $bool   $str     (      )
    +      -      *      /
    <      =      >     <=
```

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

#
    S -> PROG #1
  PROG -> CALCS #2 |
          DEFS #3 |
          DEFS CALCS #4
 CALCS -> CALC #5 |
          CALCS CALC #6
  CALC -> E #7 |
          BOOL #8 |
          STR #9 |
          DISPSET #10
     E -> $id #11 |
          $zero #12 |
          ADD #13 |
          SUB #14 |
          DIV #15 |
          MUL #16 |
          COND #17 |
          IF #18 |
          CPROC #19
   ADD -> HADD E ) #20
  HADD -> ( + #21 |
          HADD E #22
   SUB -> HSUB E ) #23
  HSUB -> ( - #24 |
          HSUB E #25
   DIV -> HDIV E ) #26
  HDIV -> ( / #27 |
          HDIV E #28
   MUL -> HMUL E ) #29
  HMUL -> ( * #30 |
          HMUL E #31
  COND -> HCOND ELSE ) #32 |
          HCOND CLAUS ) #33
 HCOND -> ( cond #34 |
          HCOND CLAUS #35
 CLAUS -> HCLAUS E ) #36
HCLAUS -> ( BOOL #37 |
          HCLAUS INTER #38
```

```
  ELSE -> HELSE E ) #39
 HELSE -> ( else #40 |
        HELSE INTER #41
   IF -> ( if BOOL E E ) #42
 CPROC -> HCPROC ) #43
HCPROC -> ( $id #44 |
        HCPROC E #45
  BOOL -> $bool #46 |
        $idq #47 |
        CPRED #48 |
        REL #49 |
        ( not BOOL ) #50
 CPRED -> HCPRED ) #51
HCPRED -> ( $idq #52 |
        HCPRED ARG #53
   ARG -> E #54 |
        BOOL #55
   REL -> ( < E E ) #56 |
        ( = E E ) #57
   STR -> $str #58
   SET -> ( set! $id E ) #59
DISPSET -> ( display E ) #60 |
        ( display BOOL ) #61 |
        ( display STR ) #62 |
        ( newline ) #63 |
        SET #64
 INTER -> DISPSET #65 |
        E #66
  DEFS -> DEF #67 |
        DEFS DEF #68
   DEF -> PRED #69 |
        VAR #70 |
        PROC #71
  PRED -> HPRED BOOL ) #72
 HPRED -> PDPAR ) #73
 PDPAR -> ( define ( $idq #74 |
        PDPAR $idq #75 |
        PDPAR $id #76
 CONST -> $zero #77 |
        $dec #78
  VAR -> ( define $id CONST ) #79
  PROC -> HPROC LET ) #80 |
```

```
        HPROC E ) #81
 HPROC -> PCPAR ) #82 |
        HPROC INTER #83
 PCPAR -> ( define ( $id #84 |
        PCPAR $id #85
   LET -> HLET E ) #86
  HLET -> LETLOC ) #87 |
        HLET INTER #88
LETLOC -> ( let ( #89 |
        LETLOC LETVAR #90
LETVAR -> ( $id E ) #91
```

**Особенности грамматики по форме GrammaFeatures.rtf .**

**>1. Вычитание.**

**1.1 Один и более операндов.**

**(- x y z)**

**2. Деление.**

**2.1 Один и более операндов.**

**(/ x y z)**

**3. Числовые литералы токена $zero.**

***3.1 В общем контексте числового выражения.**

**0**

**4. Числовые литералы токена $dec.**

**4.2 Только в определении глобальной переменной.**

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

**5. Форма or.**

**5.2 Отсутствует.**

**6. Форма and.**

**6.2 Отсутствует.**

**7. Форма not.**

**7.1 Есть.**

**(not #t )**

**8. Оператор = .**

**8.1 Есть.**

**(= x y)**

**9. Оператор отношения, кроме оператора = .**

**9.1 (< x y)**

**10. Форма IF для чисел.**

**10.1 Есть.**
**(if #t e pi)**

**11. Форма IF для строк.**

**11.2 Отсутствует.**

**12. Форма COND.**

**12.1 Ветвь ELSE и несколько клауз, а также несколько клауз без ветви ELSE.**
**(cond(x? pi)(y? e)(else 0)) (cond(x? pi)(y? e)(#t 0))**

**13. Глобальные переменные.**

**13.1 Есть**
**(define a 1)a**

**13. Локальные переменные.**

**13.1 Определяются формой let.**
**(define (f)(let((a pi))a)) (f)**

## Контрольная задача №1.

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

```
Input gramma name>var19-206/a07
Gramma:var19-206/a07.txt
Source>tests/even-odd
Source:tests/even-odd.ss
   1|;even-odd
   2|(define zero 0)
   3|(define one 1)
   4|(define two 2)
   5|(define(even-bits n)
   6| (cond((= n zero)one)
   7|  ((=(remainder n two)zero)
   8|   (even-bits (quotient n two)))
   9|  (else(odd-bits(quotient n two)))
  10| ))
  11|(define(odd-bits n)
  12| (cond((= n zero)zero)
  13|  ((=(remainder n two)zero)
  14|   (odd-bits (quotient n two)))
  15|  (#t(even-bits(quotient n two)))
  16| ))
  17|(define(display-bin n)
  18| (display(remainder n two))
  19| (if(= n zero)zero (display-bin (quotient n two)))
  20|)
  21|(define(report-results n)
  22| (display "Happy birthday to you!\n\t")
  23| (display n)(display " (decimal)\n")
  24| (display "\teven?\t")(display (if(=(even-bits n)one) one zero))
  25| (newline)
  26| (display "\todd?\t")(display (if(=(odd-bits n)one) one zero))
  27| (newline)
  28| (display "(reversed binary)\n")
  29| (let ((bin (display-bin n) ))
  30|  bin
  31|  (newline)
  32|  zero)
  33|
  34|
  35|
  36|)
  37|;***** Date of YOUR birthday *******
```

```
;***** Date of YOUR birthday *******
(define dd 23)
(define mm 3)
(define yyyy 1999)
;**********************************
(define ddnum 1000000)
(define mmnum 10000)
(report-results (+ (* dd ddnum)
                   (* mm mmnum)
                   yyyy))

```

Code:

```
/*  dao2019   */
#include "mlisp.h"
        double even__bits(      double n);
        double odd__bits(       double n);
        double display__bin(    double n);
        double report__results( double n);
//______________
        double zero = 0;
        double one = 1;
        double two = 2;
        double even__bits(      double n) {
        return  ( ( n == zero ) ? (one) : ( remainder( n, two) == zero ) ? (even__bits( q
uotient( n, two))) : ( odd__bits( quotient( n, two))) );
}
        double odd__bits(       double n) {
        return  ( ( n == zero ) ? (zero) : ( remainder( n, two) == zero ) ? (odd__bits( q
uotient( n, two))) : true ? (even__bits( quotient( n, two))) :  _infinity);
}
        double display__bin(    double n) {
        display( remainder( n, two));
        return ( ( n == zero ) ? zero : display__bin( quotient( n, two)) );
}
        double report__results( double n) {
        display( "Happy birthday to you!\n\t"); display( n);
        display( " (decimal)\n");       display( "\teven?\t");  display( ( ( even__bits(
n) == one ) ? one : zero ));
        newline();
        display( "\todd?\t");   display( ( ( odd__bits( n) == one ) ? one : zero ));
```

```
        newline();
        display( "(reversed binary)\n");        double bin = display__bin( n);
bin;
        newline();
return zero;
}
        double dd = 23;
        double mm = 3;
        double yyyy = 1999;
        double ddnum = 1000000;
        double mmnum = 10000;
int main(){
        display(report__results( ( ( dd * ddnum ) + ( mm * mmnum ) + yyyy ))); newline();

        std::cin.get();
        return 0;
}
```

**Протокол запуска задачи на C++.**
**>**

```
art@mars:~/study/semester_4/SP/mylabs/kp/curs1/tests$ g++ even-odd.cpp
art@mars:~/study/semester_4/SP/mylabs/kp/curs1/tests$ ./a.out
Happy birthday to you!
        23031999 (decimal)
        even?   0
        odd?    1
(reversed binary)
11111101000011101111101010
0
```

**Протокол запуска задачи на Лиспе.**
**>**

```
Добро пожаловать в DrRacket, версия 7.2 [3m].
Язык: Pretty Big; memory limit: 128 MB.
Happy birthday to you!
        23031999 (decimal)
        even?   0
        odd?    1
(reversed binary)
11111101000011101111101010
0
> |
```

**Контрольная задача №2.**

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

```
Input gramma name>var19-206/a07
Gramma:var19-206/a07.txt
Source>tests/golden-section
Source:tests/golden-section.ss
   1|;golden-section
   2|
   3|(define a 19)
   4|(define b 20);19.999995177562162
   5|
   6|(define one-zero-seven 107)
   7|(define one-zero-eight 108)
   8|(define zero-point-one 0.1)
   9|(define two 2)
  10|(define zero-point-five 0.5)
  11|(define one 1)
  12|(define five 5)
  13|(define mphi 1.61803398875)
  14|
  15|
  16|(define (fun x)
  17| (let(( y (- x (/ one-zero-seven one-zero-eight e))))
  18|    (-
  19|      (* zero-point-one (expt y two))
  20|      (* y (log y))
  21|    )
  22| )
  23|;     0.1x^2 - xlnx
  24|)
  25|(define eps 0.00001)
  26|
  27|(define (golden-section-search a b)
  28| (let(
  29|      (xmin(if(< a b)(golden-start a b)(golden-start b a )))
  30|     )
  31|     (newline)
  32|     xmin
  33| )
  34|)
  35|(define (golden-start a b)
  36|
```

```
 (let(
      (x-one (- b (/ (- b a) mphi)))
      (x-two (+ a (/ (- b a) mphi)))
     )
     (try a b x-one (fun x-one) x-two (fun x-two))
 )
)


(define (try a b x-one y-one x-two y-two)
 (cond((<(abs (- a b))eps)(* (+ a b)zero-point-five))
      (#t (display "+")
          (cond((< y-one y-two)(set! b x-two)
                          (set! x-two (+ a (/ (- b a) mphi)))
                          (set! x-one (- b (/ (- b a) mphi)))
                          (try a b x-one (fun x-one) x-two (fun x-two))
                )
                (else    (set! a x-one)
                          (set! x-one (- b (/ (- b a) mphi)))
                          (set! x-two (+ a (/ (- b a) mphi)))
                          (try a b x-one (fun x-one) x-two (fun x-two))
                )
          )
       )
 )
)

(define xmin 0)
(set! xmin(golden-section-search a b))
  (display"interval=\t[")
  (display a)
  (display" , ")
  (display b)
  (display"]\n")
  (display"xmin=\t\t")
xmin
  (display"f(xmin)=\t")
(fun xmin)

```

Code:

```
/*  dao2019   */
#include "mlisp.h"
        double fun(     double x);
        double golden__section__search( double a,       double b);
        double golden__start(   double a,       double b);
        double __dao2019__try(  double a,       double b,       double x__one,  double y_
_one,  double x__two,  double y__two);
//________________
        double a = 19;
        double b = 20;
        double one__zero__seven = 107;
        double one__zero__eight = 108;
        double zero__point__one = 0.1;
        double two = 2;
        double zero__point__five = 0.5;
        double one = 1;
        double five = 5;
        double mphi = 1.61803398875;
        double fun(     double x) {
        double y = ( x - ( one__zero__seven / one__zero__eight / e ) );
return ( ( zero__point__one * expt( y, two) ) - ( y * log( y) ) );
}
        double eps = 0.00001;
        double golden__section__search( double a,       double b) {
        double xmin = ( ( a < b ) ? golden__start( a, b) : golden__start( b, a) );;
        newline();
return xmin;
}
        double golden__start(   double a,       double b) {
        double x__one = ( b - ( ( b - a ) / mphi ) );
        double x__two = ( a + ( ( b - a ) / mphi ) );
return __dao2019__try( a, b, x__one, fun( x__one), x__two, fun( x__two));
}
        double __dao2019__try(  double a,       double b,       double x__one,  double y_
_one,  double x__two,  double y__two) {
        return  ( ( abs( ( a - b )) < eps ) ? (( ( a + b ) * zero__point__five )) : true
? (    display( "+"),  ( ( y__one < y__two ) ? (b = x__two, x__two = ( a + ( ( b - a ) /
mphi ) ), x__one = ( b - ( ( b - a ) / mphi ) ), __dao2019__try( a, b, x__one, fun( x__on
e), x__two, fun( x__two))) : ( a = x__one, x__one = ( b - ( ( b - a ) / mphi ) ), x__two
= ( a + ( ( b - a ) / mphi ) ), __dao2019__try( a, b, x__one, fun( x__one), x__two, fun(
x  two))) )) :   infinity);
```

```
}
        double xmin = 0;
int main(){
xmin = golden__section__search( a, b);

        display( "interval=\t[");
        display( a);

        display( " , ");
        display( b);

        display( "]\n");
        display( "xmin=\t\t");
        display(xmin); newline();
        display( "f(xmin)=\t");
        display(fun( xmin)); newline();

        std::cin.get();
        return 0;
}
```

**Протокол запуска задачи на C++.**

**>**

```
art@mars:~/study/semester_4/SP/mylabs/kp/curs1/tests$ g++ golden-section.cpp

art@mars:~/study/semester_4/SP/mylabs/kp/curs1/tests$ ./a.out
++++++++++++++++++++++
interval=       [19 , 20]
xmin=           19.99999517756216
f(xmin)=        -19.90625800928606
```

**Протокол запуска задачи на Лиспе.**

```
Добро пожаловать в DrRacket, версия 7.2 [3m].
Язык: Pretty Big; memory limit: 128 MB.
++++++++++++++++++++++++
interval=       [19 , 20]
xmin=           19.999995177562162
f(xmin)=        -19.906258009286063
> |
```

**>Контрольная задача №3.**

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

```
Source>tests/coin19
Source:tests/coin19.ss
  1|(define variant 7)
  2|(define last-digit-of-group-number 6)
  3|(define largest-coin 50)
  4|
  5|(define one 1)
  6|(define zero 0)
  7|
  8|(define (cc amount largest-coin)
  9| (cond
 10|
 11|
 12|  ((= (cond
 13|       ((not (eq? largest-coin one))
 14|        (cond
 15|              ((not (eq? amount one)) zero)
 16|              (else one)
 17|        )
 18|       )
 19|       (else one)
 20|      ) one) one)
 21|
 22|
 23|
 24|  ((= (cond
 25|     ((not (< amount zero))
 26|      (cond ((not (eq? largest-coin zero)) zero)
 27|        (else one)
 28|      )
 29|     )
 30|     (else one)
 31|    ) one) zero)
 32|
 33|
 34|  (else (+ (cc amount (next-coin largest-coin)) (cc (- amount largest-coin) largest-
coin)))
 35| )
 36|)
 37|
 38|(define (count-change amount)
```

```
 39| (cc amount largest-coin)
 40|)
 41|
 42|
 43|(define ten 10)
 44|(define five 5)
 45|(define three 3)
 46|(define two 2)
 47|(define one-zero-zero 100)
 48|(define one-three-seven 137)
 49|
 50|(define (next-coin coin)
 51| (cond ((eq? coin largest-coin) ten)
 52|  ((eq? coin ten) five)
 53|  ((eq? coin five) three)
 54|  ((eq? coin three) two)
 55|  ((eq? coin two) one)
 56|  (else zero)
 57| )
 58|)
 59|
 60|(define (gr-amount)
 61| (remainder (+ (* one-zero-zero last-digit-of-group-number) variant) one-three-seven
)
 62|)
 63|
 64|(display "dao variant ")
 65|(display variant)
 66|(newline)
 67|(display "1-2-3-5-10-50")
 68|(newline)
 69|(display "count__change for 100 \t= ")
 70|(display (count-change one-zero-zero))
 71|(newline)
 72|(display "count__change for ")
 73|(display (gr-amount))
 74|(display " \t= ")
 75|(display (count-change (gr-amount)))
 76|(newline)
 77|
```

```
_________________
Code:
/*  dao2019   */
#include "mlisp.h"
        double cc(      double amount,  double largest__coin);
        double count__change(   double amount);
        double next__coin(      double coin);
        double gr__amount();
//_________________
        double variant = 7;
        double last__digit__of__group__number = 6;
        double largest__coin = 50;
        double one = 1;
        double zero = 0;
        double cc(      double amount,  double largest__coin) {
        return  ( (  ( (!eq_Q( largest__coin, one)) ? ( ( (!eq_Q( amount, one)) ? (zero)
: ( one) )) : ( one) ) == one ) ? (one) : (  ( (!( amount < zero )) ? ( ( (!eq_Q( largest
__coin, zero)) ? (zero) : ( one) )) : ( one) ) == one ) ? (zero) : ( ( cc( amount, next__
coin( largest__coin)) + cc( ( amount - largest__coin ), largest__coin) )) );
}
        double count__change(   double amount) {
        return cc( amount, largest__coin);
}
        double ten = 10;
        double five = 5;
        double three = 3;
        double two = 2;
        double one__zero__zero = 100;
        double one__three__seven = 137;
        double next__coin(      double coin) {
        return  ( eq_Q( coin, largest__coin) ? (ten) : eq_Q( coin, ten) ? (five) : eq_Q(
coin, five) ? (three) : eq_Q( coin, three) ? (two) : eq_Q( coin, two) ? (one) : ( zero) )
;
}
        double gr__amount() {
        return remainder( ( ( one__zero__zero * last__digit__of__group__number ) + varian
t ), one__three__seven);
}
int main(){
        display( "dao variant ");
        display( variant);

        newline();

        display( "1-2-3-5-10-50");
        newline();

        display( "count__change for 100 \t= ");
        display( count__change( one__zero__zero));

        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++.**
**>**

```
art@mars:~/study/semester_4/SP/mylabs/kp/curs1/tests$ g++ coin19.cpp
art@mars:~/study/semester_4/SP/mylabs/kp/curs1/tests$ ./a.out
dao variant 7
1-2-3-5-10-50
count_change for 100   = 22420
count_change for 59    = 3208
```

**Протокол запуска задачи на Лиспе.**
**>**

```
Добро пожаловать в DrRacket, версия 7.2 [3m].
Язык: Pretty Big; memory limit: 128 MB.
dao variant 7
1-2-3-5-10-50
count_change for 100   = 22420
count_change for 59    = 3208
>
```

**Распечатка файла code-gen.cpp.**

```
>/* $a07 */
#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 -> CALCS
   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(){\n"
           "\tdisplay(\"No calculations!\");newline();\n"
           "\tstd::cin.get();\n\treturn 0;\n}\n";
   return 0;
}

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

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

int tCG::p06() { //  CALCS -> CALCS CALC
   S1->obj += S2->obj;
   return 0;
}

int tCG::p07() { //   CALC -> E
   /*if (S1->obj[S1->obj.size()-1] == ';') {
      S1->obj = S1->obj.substr(0,S3->obj.size()-1);
   }*/
   S1->obj = "\tdisplay(" + S1->obj + "); newline();\n";
   return 0;
}

int tCG::p08() { //   CALC -> BOOL
   /*if (S1->obj[S1->obj.size()-1] == ';') {
      S1->obj = S1->obj.substr(0,S1->obj.size()-1);
   }*/
   S1->obj = "\tdisplay(" + S1->obj + "); newline();\n";
   return 0;
}

int tCG::p09() { //   CALC -> STR
   /*if (S1->obj[S1->obj.size()-1] == ';') {
      S1->obj = S1->obj.substr(0,S1->obj.size()-1);
   }*/
```

```
   S1->obj = "\tdisplay(" + S1->obj + "); newline();\n";
   return 0;
}

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

int tCG::p11() { //     E -> $id
   S1->obj = decor(S1->name);
   return 0;
}

int tCG::p12() { //     E -> $zero
   S1->obj = decor(S1->name);
   return 0;
}

int tCG::p13() { //     E -> ADD
   S1->obj = "( " + S1->obj + " )";
   return 0;
}

int tCG::p14() { //     E -> SUB
   S1->obj = "( " + S1->obj + " )";
   return 0;
}

int tCG::p15() { //     E -> DIV
   S1->obj = "( " + S1->obj + " )";
   return 0;
}

int tCG::p16() { //     E -> MUL
   S1->obj = "( " + S1->obj + " )";
   return 0;
}

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

int tCG::p18() { //     E -> IF
```

```
   return 0;
}

int tCG::p19() { //     E -> CPROC
   /*if (S1->obj[S1->obj.size()-1] == ';') {
      S1->obj = S1->obj.substr(0,S1->obj.size()-1);
   }*/
   return 0;
}

int tCG::p20() { //    ADD -> HADD E )
   if (S1->count == 0) //один операнд
      S1->obj = S2->obj;
   else          //более одного операнда
      S1->obj += S2->obj;
   S1->count = 0;
   return 0;
}

int tCG::p21() { //   HADD -> ( +
   return 0;
}

int tCG::p22() { //   HADD -> HADD E
   S1->obj += S2->obj + " + ";
   ++S1->count;
   return 0;
}

int tCG::p23() { //    SUB -> HSUB E )
   if (S1->count == 0) //один операнд
      S1->obj = S2->obj;
   else          //более одного операнда
      S1->obj += S2->obj;
   S1->count = 0;
   return 0;
}

int tCG::p24() { //   HSUB -> ( -
   return 0;
}

int tCG::p25() { //   HSUB -> HSUB E
   S1->obj += S2->obj + " - ";
```

```
   ++S1->count;
   return 0;
}

int tCG::p26() { //    DIV -> HDIV E )
   if (S1->count == 0) //один операнд
      S1->obj = S2->obj;
   else          //более одного операнда
      S1->obj += S2->obj;
   S1->count = 0;
   return 0;
}

int tCG::p27() { //   HDIV -> ( /
   return 0;
}

int tCG::p28() { //   HDIV -> HDIV E
   S1->obj += S2->obj + " / ";
   ++S1->count;
   return 0;
}

int tCG::p29() { //    MUL -> HMUL E )
   if (S1->count == 0) //один операнд
      S1->obj = S2->obj;
   else          //более одного операнда
      S1->obj += S2->obj;
   S1->count = 0;
   return 0;
}

int tCG::p30() { //   HMUL -> ( *
   return 0;
}

int tCG::p31() { //   HMUL -> HMUL E
   S1->obj += S2->obj + " * ";
   ++S1->count;
   return 0;
}

// COND = (cond CLAUS ELSE) = (CLAUS CLAUS ELSE);
int tCG::p32() { //   COND -> HCOND ELSE )
```

```
   if (S2->obj[S2->obj.size()-1] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-1);
   }
   else if (S2->obj[S2->obj.size()-1] == '\n' && S2-
>obj[S2->obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }
   S1->obj = S1->obj + S2->obj + " ); ";
   return 0;
}

// COND = (cond CLAUS CLAUS) = (CLAUS CLAUS _infinity);
int tCG::p33() { //   COND -> HCOND CLAUS )
   if (S2->obj[S2->obj.size()-1] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-1);
   }
   else if (S2->obj[S2->obj.size()-1] == '\n' && S2-
>obj[S2->obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }

   S1->obj = S1->obj + S2->obj + " _infinity); ";
   return 0;
}

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

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

// CLAUS = (BOOL E) = BOOL ? E :
int tCG::p36() { //  CLAUS -> HCLAUS E )

   if (S2->obj[S2->obj.size()-1] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-1);
   }
```

```
   else if (S2->obj[S2->obj.size()-1] == '\n' && S2-
>obj[S2->obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }
   else if (S2->obj[S2->obj.size()-1] == ' ' && S2->obj[S2-
>obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }


   if(S1->count)
      S1->obj += ", " + S2->obj + ") : ";
   else
      S1->obj += S2->obj + ") : ";
   S1->count = 0;


   return 0;
}

int tCG::p37() { // HCLAUS -> ( BOOL
   S1->obj = S2->obj + " ? (";
   return 0;
}

int tCG::p38() { // HCLAUS -> HCLAUS INTER
   if (S2->obj[S2->obj.size()-1] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-1);
   }
   else if (S2->obj[S2->obj.size()-1] == '\n' && S2-
>obj[S2->obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }

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

// ELSE = (else E) = (E)
int tCG::p39() { //   ELSE -> HELSE E )
```

```
   if (S2->obj[S2->obj.size()-1] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-1);
   }
   else if (S2->obj[S2->obj.size()-1] == '\n' && S2-
>obj[S2->obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }


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

int tCG::p40() { //  HELSE -> ( else
   S1->obj = "( ";
   return 0;
}

int tCG::p41() { //  HELSE -> HELSE INTER
   if (S2->obj[S2->obj.size()-1] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-1);
   }
   else if (S2->obj[S2->obj.size()-1] == '\n' && S2-
>obj[S2->obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }
   if(S1->count)
      S1->obj += ", " + S2->obj;
   else
      S1->obj += S2->obj;
   S1->count++;
   return 0;
}

// IF = (if BOOl E E) = (BOOL ? E : E)
int tCG::p42() { //     IF -> ( if BOOL E E );
   S1->obj = "( " + S3->obj + " ? " + S4->obj + " : " + S5-
>obj + " );";
   return 0;
```

```
}

int tCG::p43() { //  CPROC -> HCPROC )
   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 += ", ";// не первый параметр
   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 -> $idq
   S1->obj = decor(S1->name);
   return 0;
}

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

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

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

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

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

int tCG::p53() { // HCPRED -> HCPRED ARG
   if (S1->count)
      S1->obj += ", ";// не первый параметр
   S1->obj += S2->obj;
   ++(S1->count);
   return 0;
}

int tCG::p54() { //    ARG -> E
   return 0;
}

int tCG::p55() { //    ARG -> BOOL
   return 0;
}

int tCG::p56() { //    REL -> ( < E E )
   S1->obj = "( " + S3->obj + " < " + S4->obj + " )";
   return 0;
}

int tCG::p57() { //    REL -> ( = E E )
   if (S3->obj[S3->obj.size()-1] == ';') {
      S3->obj = S3->obj.substr(0,S3->obj.size()-1);
   }
   else if (S3->obj[S3->obj.size()-1] == '\n' && S3-
>obj[S3->obj.size()-2] == ';') {
      S3->obj = S3->obj.substr(0,S3->obj.size()-2);
   }
   else if (S3->obj[S3->obj.size()-1] == ' ' && S3->obj[S3-
>obj.size()-2] == ';') {
      S3->obj = S3->obj.substr(0,S3->obj.size()-2);
   }
```

```
   if (S4->obj[S4->obj.size()-1] == ';') {
      S4->obj = S4->obj.substr(0,S4->obj.size()-1);
   }
   else if (S4->obj[S4->obj.size()-1] == '\n' && S4-
>obj[S4->obj.size()-2] == ';') {
      S4->obj = S4->obj.substr(0,S4->obj.size()-2);
   }
   else if (S4->obj[S4->obj.size()-1] == ' ' && S4->obj[S4-
>obj.size()-2] == ';') {
      S4->obj = S4->obj.substr(0,S4->obj.size()-2);
   }

   S1->obj = "( " + S3->obj + " == " + S4->obj + " )";
   return 0;
}

int tCG::p58() { //    STR -> $str
   S1->obj = S1->name;
   return 0;
}

int tCG::p59() { //    SET -> ( set! $id E )
   if (S4->obj[S4->obj.size()-1] == ';') {
      S4->obj = S4->obj.substr(0,S4->obj.size()-1);
   }
   else if (S4->obj[S4->obj.size()-1] == '\n' && S4-
>obj[S4->obj.size()-2] == ';') {
      S4->obj = S4->obj.substr(0,S4->obj.size()-2);
   }
   else if (S4->obj[S4->obj.size()-1] == ' ' && S4->obj[S4-
>obj.size()-2] == ';') {
      S4->obj = S4->obj.substr(0,S4->obj.size()-2);
   }
   S1->obj = decor(S3->name) + " = " + S4->obj + ";\n";
   return 0;
}

int tCG::p60() { //DISPSET -> ( display E )
   if (S3->obj[S3->obj.size()-1] == ';') {
      S3->obj = S3->obj.substr(0,S3->obj.size()-1);
   }
   S1->obj = "\tdisplay( " + S3->obj + ");\n";
   return 0;
```

```
}

int tCG::p61() { //DISPSET -> ( display BOOL )
   if (S3->obj[S3->obj.size()-1] == ';') {
      S3->obj = S3->obj.substr(0,S3->obj.size()-1);
   }
   S1->obj = "\tdisplay( " + S3->obj + ");\n";
   return 0;
}

int tCG::p62() { //DISPSET -> ( display STR )
   if (S3->obj[S3->obj.size()-1] == ';') {
      S3->obj = S3->obj.substr(0,S3->obj.size()-1);
   }
   S1->obj = "\tdisplay( " + S3->obj + ");";
   return 0;
}

int tCG::p63() { //DISPSET -> ( newline )
   S1->obj = "\tnewline();\n";
   return 0;
}

int tCG::p64() { //DISPSET -> SET
   return 0;
}

int tCG::p65() { //  INTER -> DISPSET
   return 0;
}

int tCG::p66() { //  INTER -> E
   S1->obj = S1->obj;
   return 0;
}

int tCG::p67() { //   DEFS -> DEF
   return 0;
}

int tCG::p68() { //   DEFS -> DEFS DEF
   S1->obj+=S2->obj;
   return 0;
}
```

```
int tCG::p69() { //    DEF -> PRED
   return 0;
}

int tCG::p70() { //    DEF -> VAR
   return 0;
}

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

//
int tCG::p72() { //   PRED -> HPRED BOOL )
   S1->obj += S2->obj + ";\n}\n";
   return 0;
}

int tCG::p73() { //  HPRED -> PDPAR )
   S1->obj += ")";
   declarations += S1->obj + ";\n"; //!!!
   S1->obj += "{\n\treturn ";
   S1->count = 0;
   return 0;
}

int tCG::p74() { //  PDPAR -> ( define ( $idq
   S1->obj = "\tbool " + decor(S4->name) + "(";
   S1->count = 0;
   return 0;
}

int tCG::p75() { //  PDPAR -> PDPAR $idq
   if (S1->count)S1->obj += ", ";// не первый параметр
   S1->obj += "\tdouble " + decor(S2->name);
   ++(S1->count);
   return 0;
}

int tCG::p76() { //  PDPAR -> PDPAR $id
   if (S1->count)S1->obj += ", ";// не первый параметр
   S1->obj += "\tdouble " + decor(S2->name);
   ++(S1->count);
```

```
   return 0;
}

int tCG::p77() { //  CONST -> $zero
   S1->obj = decor(S1->name);
   return 0;
}

int tCG::p78() { //  CONST -> $dec
   S1->obj = decor(S1->name);
   return 0;
}

int tCG::p79() { //    VAR -> ( define $id CONST )
   S1->obj = "\tdouble " + decor(S3->name) + " = " + S4-
>obj + ";\n";
   return 0;
}

int tCG::p80() { //   PROC -> HPROC LET )
   if (S2->obj[S2->obj.size()-1] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-1);
   }
   else if (S2->obj[S2->obj.size()-1] == '\n' && S2-
>obj[S2->obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }
   else if (S2->obj[S2->obj.size()-1] == ' ' && S2->obj[S2-
>obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }


   S1->obj = S1->obj + S2->obj + ";\n}\n";
   return 0;
}

int tCG::p81() { //   PROC -> HPROC E )

   if (S2->obj[S2->obj.size()-1] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-1);
   }
   else if (S2->obj[S2->obj.size()-1] == '\n' && S2-
>obj[S2->obj.size()-2] == ';') {
```

```
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }
   else if (S2->obj[S2->obj.size()-1] == ' ' && S2->obj[S2-
>obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }

   S1->obj = S1->obj + "\treturn " + S2->obj + ";\n}\n";
   return 0;
}

int tCG::p82() { //  HPROC -> PCPAR )
   S1->obj += ")";
   declarations += S1->obj + ";\n"; //!!!
   S1->obj += " {\n  ";
   S1->count = 0;
   return 0;
}

int tCG::p83() { //  HPROC -> HPROC INTER
   S1->obj+=S2->obj;
   return 0;
}

int tCG::p84() { //  PCPAR -> ( define ( $id
   S1->obj = "\tdouble " + decor(S4->name) + "(";
   S1->count = 0;
   return 0;
}

int tCG::p85() { //  PCPAR -> PCPAR $id
   if (S1->count)
      S1->obj += ", ";// не первый параметр
   S1->obj += "\tdouble " + decor(S2->name);
   ++(S1->count);
   return 0;
}

// LET = ( let (LETVAR) INTER ) = LETVAR\n INTER
int tCG::p86() { //    LET -> HLET E )
   S1->obj+= "return " + S2->obj;
   return 0;
}
```

```
int tCG::p87() { //   HLET -> LETLOC )
   return 0;
}

int tCG::p88() { //   HLET -> HLET INTER
   if (S2->obj[S2->obj.size()-1] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-1);
   }
   else if (S2->obj[S2->obj.size()-1] == '\n' && S2-
>obj[S2->obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }
   else if (S2->obj[S2->obj.size()-1] == ' ' && S2->obj[S2-
>obj.size()-2] == ';') {
      S2->obj = S2->obj.substr(0,S2->obj.size()-2);
   }
   S1->obj+=  S2->obj + ";\n";
   return 0;
}

int tCG::p89() { // LETLOC -> ( let (
   return 0;
}

int tCG::p90() { // LETLOC -> LETLOC LETVAR
   S1->obj+=S2->obj + "\n";
   return 0;
}

// double $id = E;
int tCG::p91() { // LETVAR -> ( $id E )
   S1->obj = "\tdouble " + decor(S2->name) + " = " + S3-
>obj + ";";
   return 0;
}

//_____________________________
int tCG::p92() { return 0; }

int tCG::p93() { return 0; }

int tCG::p94() { 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; }**

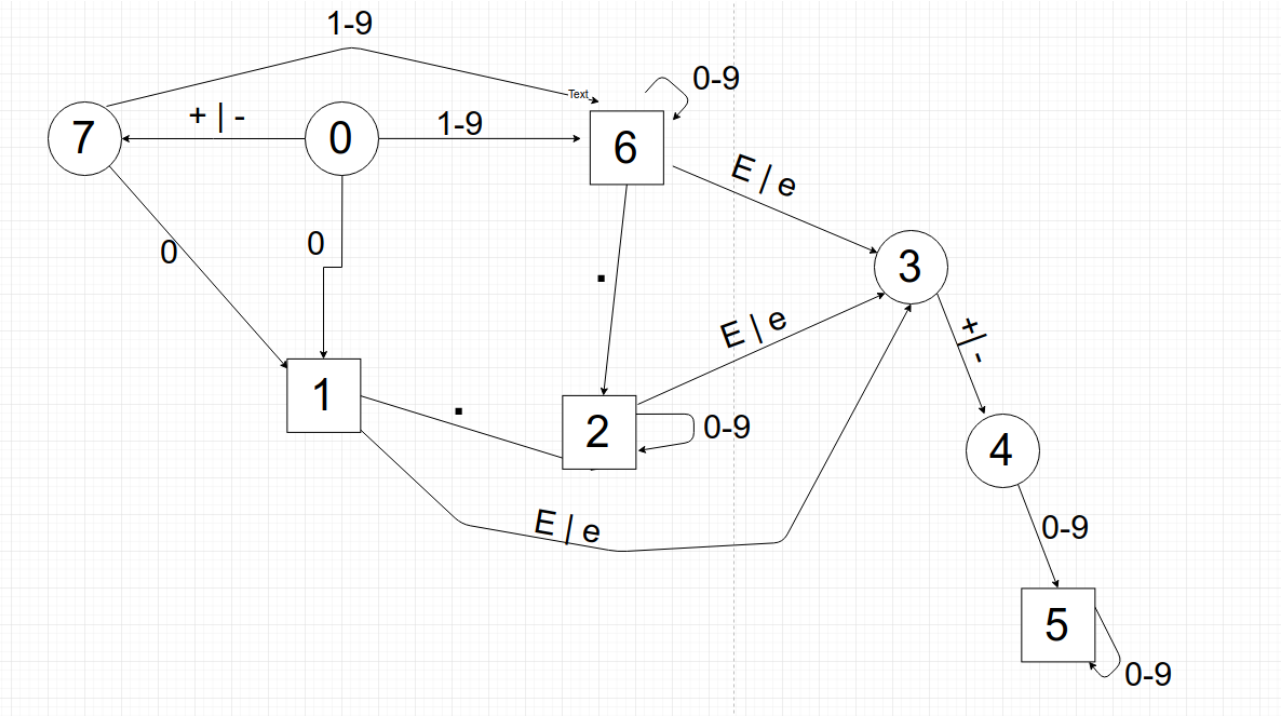
**int tCG::p107() { return 0; }**

**int tCG::p108() { return 0; }**

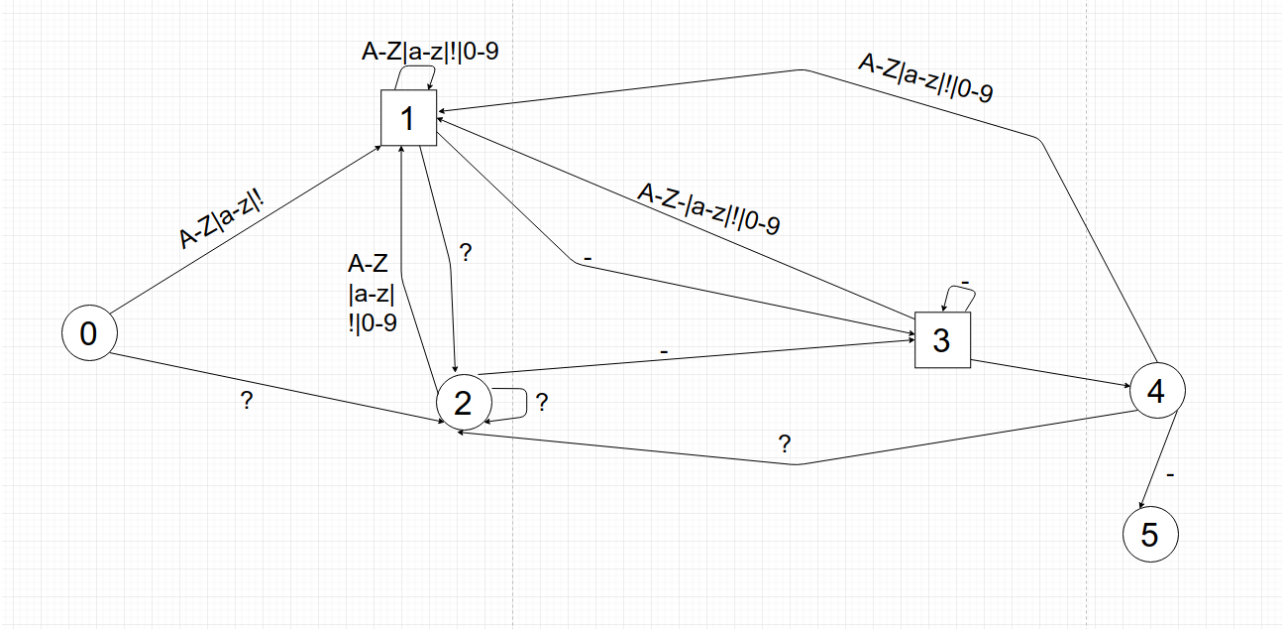
**int tCG::p109() { return 0; }**

**int tCG::p110() { return 0; }**

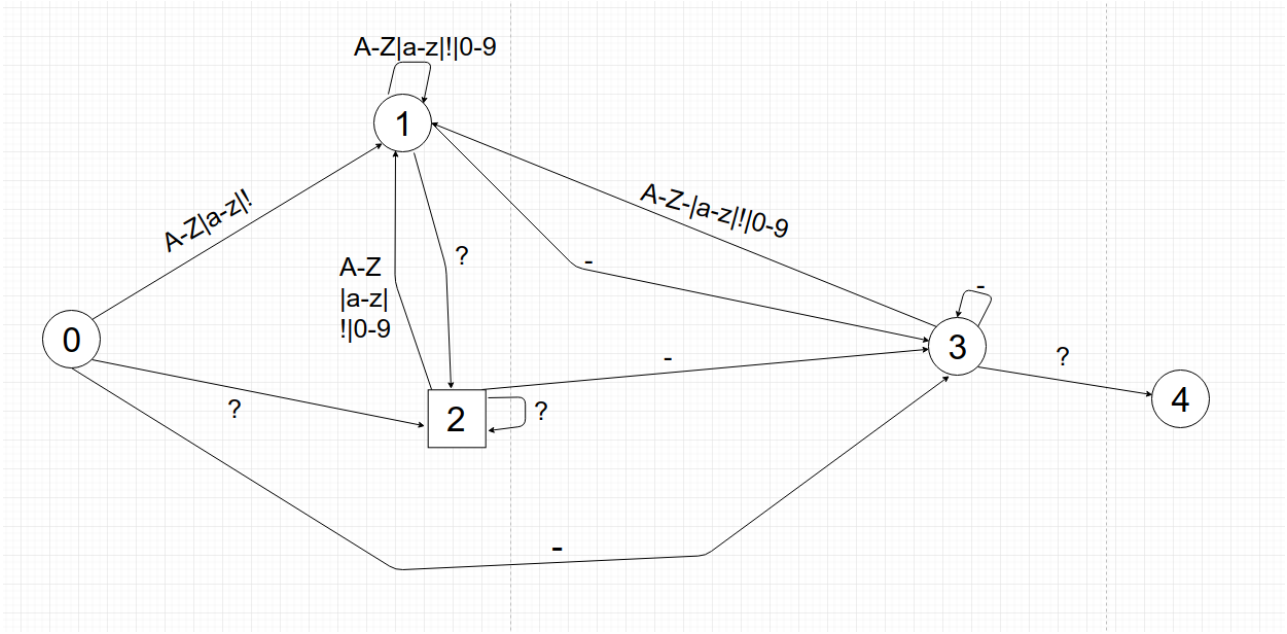
**Диаграммы автоматов из лабораторной работы №5 для токенов $dec, $id, $idq. Над каждой диаграммой проставить номер варианта шаблона токена и его краткое описание. Все диаграммы должны быть построены в одном редакторе и должны иметь единый стиль изображения.**
**>**
**$dec: 3. Дробную часть можно опустить, СОХРАНЯЯ точку, например, 0., 1.e+7**




**$id: Запрещенные символы (-?-)**




**$idq: Запрещенные символы (-?)**




**Выводы по проделанной работе - не менее одной страницы не разбавленного «водой» текста.**
**>Проделав лабораторную работу, я изучил и разработал синтаксически управляемый транслятор (генератор кода) в язык C++ из языка MicroLisp. Для этого, согласно заданной грамматике, мной был создан code-gen.cpp. Для того чтобы лучше разобраться в процессе создания транслятора, мной сперва были выполнены лабораторные работы 8 и 9, в которых были рассмотрены отдельные случаи, которые могут возникнуть в процессе трансляции. Согласно синтаксическим продукциям, по частям собирается код на C++. Поскольку грамматика, выданная по заданию, обладает довольно нестандартными ограничениями, например, присутствие операторов < , =, and и or , контрольные задачи пришлось адаптировать к заданному диалекту языка, что вылилось в громоздкие формулы. Хотя верная работа адаптированных мной формул была проверена не раз.**

**Определенные затруднения вызвало написание составных объектов, например: согласно правилам трансляции, элементы составной ветви должны быть разделены оператором C++ ",", в то же время они могут, очевидно, использоваться и не как элементы ветви cond, при этом после них, согласно синтаксису языка C++, должна идти ";". Для решения этого противоречия, я при**

**закрывании составных частей cond проверял внутренний элемент который в него входит и если там присутствовал ";", то удалял его.**

**Написание транслятора осуществлялось не сразу для контрольных примеров. Сперва я воспользовался наработками из предыдущих лабораторных работ как фундаментом для создаваемого транслятора, т.е. использовал уже созданные элементы кода. Затем я обработал примеры особенностей грамматики, это дало мне дало возможность транслировать большую часть контрольной задачи. После, была доведена до конца трансляция задачи coin19.ss, так как она оказалась наиболее простой в плане используемых конструкций.**

**Транслятор прекрасно себя показал на контрольных задачах и примерах. Курсовую работу осуществил в полном объёме.**