**Студент: Куликов А.В.**
**Группа: 208**
**Номер по списку: 9**

**«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.**

```
>
# $b09
   $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 |
           CPROC #18
    ADD -> HADD E ) #19
   HADD -> ( + #20 |
           HADD E #21
    SUB -> HSUB E ) #22
   HSUB -> ( - #23 |
           HSUB E #24
    DIV -> HDIV E ) #25
   HDIV -> ( / #26 |
           HDIV E #27
    MUL -> HMUL E ) #28
   HMUL -> ( * #29 |
           HMUL E #30
   COND -> HCOND CLAUS ) #31
  HCOND -> ( cond #32 |
           HCOND CLAUS #33
  CLAUS -> HCLAUS E ) #34
 HCLAUS -> ( BOOL #35 |
           HCLAUS INTER #36
   ELSE -> HELSE E ) #37
  HELSE -> ( else #38 |
           HELSE INTER #39
  CPROC -> HCPROC ) #40
```

```
 HCPROC -> ( $id #41 |
        HCPROC E #42
  BOOL -> $bool #43 |
        $idq #44 |
        CPRED #45 |
        REL #46
 CPRED -> HCPRED ) #47
 HCPRED -> ( $idq #48 |
        HCPRED ARG #49
   ARG -> E #50 |
        BOOL #51
   REL -> ( < E E ) #52 |
        ( = E E ) #53
   STR -> $str #54 |
        SIF #55
   SIF -> ( if BOOL STR STR ) #56
   SET -> ( set! $id E ) #57
DISPSET -> ( display E ) #58 |
        ( display BOOL ) #59 |
        ( display STR ) #60 |
        ( newline ) #61 |
        SET #62
 INTER -> DISPSET #63 |
        E #64
  DEFS -> DEF #65 |
        DEFS DEF #66
   DEF -> PRED #67 |
        VAR #68 |
        PROC #69
  PRED -> HPRED BOOL ) #70
 HPRED -> PDPAR ) #71
 PDPAR -> ( define ( $idq #72 |
        PDPAR $idq #73 |
        PDPAR $id #74
 CONST -> $zero #75 |
        $dec #76
  VAR -> ( define $id CONST ) #77
 PROC -> HPROC LET ) #78 |
        HPROC E ) #79
 HPROC -> PCPAR ) #80 |
        HPROC INTER #81
 PCPAR -> ( define ( $id #82 |
        PCPAR $id #83
  LET -> HLET E ) #84
```

```
  HLET -> LETLOC ) #85 |
          HLET INTER #86
LETLOC -> ( let ( #87 |
          LETLOC LETVAR #88
LETVAR -> ( $id E ) #89
```

**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. Форма not.**
**7.1 Есть.**
**(not #t )**
***7.2 Отсутствует.**

**8. Оператор  = .**

***8.1 Есть.**

**(=  x y)**

**8.2 Отсутствует.**

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

***9.1 (<  x y)**

**9.2 (<= x y)**

**9.3 (>  x y)**

**9.4 (>= x y)**

**10. Форма IF для чисел.**

**10.1 Есть.**

**(if #t e pi)**

***10.2 Отсутствует.**

**11. Форма IF для строк.**

***11.1 Есть.**

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

**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))**

**12.2  Ветвь ELSE и несколько клауз.**

**(cond(x? pi)(y? e)(else 0))**

***12.3  Несколько клауз без ветви ELSE.**

**(cond(x? pi)(y? e)(#t 0))**

**12.4 Ветвь ELSE, одна клауза.**

**(cond((p?)e)(else 0))**

**12.5 Без ветви ELSE, две  клаузы.**

**(cond((p?)e)(#t 0))**

**13. Глобальные переменные.**

***13.1 Есть**

**(define a 1)a**

**13.2 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).**

>

```
Input gramma name>b09
Gramma:b09.txt
Source>even-odd
Source:even-odd.ss
   1|;even-odd
   2|
   3|(define var1 1)
   4|(define var2 2)
   5|(define var1000000 1000000)
   6|(define var10000 10000)
   7|
   8|
   9|(define(even-bits n)
  10|  (cond((= n 0)var1)
  11|        ((=(remainder n var2)0)
  12|           (even-bits (quotient n var2)))
  13|        (#t (odd-bits(quotient n var2)))
  14|        ))
  15|(define(odd-bits n)
  16|  (cond((= n 0)0)
  17|        ((=(remainder n var2)0)
  18|           (odd-bits (quotient n var2)))
  19|        (#t (even-bits(quotient n var2)))
  20|        ))
  21|(define(display-bin n)
  22|  (display(remainder n var2))
  23|  (cond ((= n 0) 0) (#t(display-bin (quotient n var2))))
  24|        )
  25|(define(report-results n)
  26|  (display "Happy birthday to you!\n\t")
  27|  (display n)(display " (decimal)\n\t")
  28|  (display-bin n)(display "(reversed binary)\n")
  29|  (display "\teven?\t")(display (if(=(even-bits n)var1) "yes" "no"))
  30|  (newline)
  31|  (display "\todd?\t")(display (if(=(odd-bits n)var1) "yes" "no"))
  32|  (newline)
  33|  0
  34|        )
  35|;***** Date of YOUR birthday *******
  36|(define dd 29)
  37|(define mm 4)
  38|(define yyyy 1999)
  39|;***********************************
  40|(report-results (+ (* dd var1000000)
```

```
  41|                   (* mm var10000)
  42|                   yyyy))
  43|
  44|
  45|
  46|
```

_____

Code:

```
/*  KAV2019   */
#include "mlisp.h"

double var1 = 1;
double var2 = 2;
double var1000000 = 1000000;
double var10000 = 10000;
double dd = 29;
double mm = 4;
double yyyy = 1999;

double even__bits(double n);
double odd__bits(double n);
double display__bin(double n);
double report__results(double n);
//_____________
double even__bits(double n){
    return (((n == 0) ? var1
        : ((remainder(n, var2) == 0) ? even__bits(quotient(n, var2))
            : (true ? odd__bits(quotient(n, var2))
                : _infinity))));
}

double odd__bits(double n){
    return (((n == 0) ? 0
        : ((remainder(n, var2) == 0) ? odd__bits(quotient(n, var2))
            : (true ? even__bits(quotient(n, var2))
                : _infinity))));
}

double display__bin(double n){
    display(remainder(n, var2));
    return (((n == 0) ? 0
        : (true ? display__bin(quotient(n, var2))
```

```
            : _infinity)));
}

double report__results(double n){
    display("Happy birthday to you!\n\t");
    display(n);
    display(" (decimal)\n\t");
    display__bin(n);
    display("(reversed binary)\n");
    display("\teven?\t");
    display(((even__bits(n) == var1) ? "yes" : "no"));
    newline();
    display("\todd?\t");
    display(((odd__bits(n) == var1) ? "yes" : "no"));
    newline();
    return 0;
}

int main(){
    display(report__results(((dd * var1000000) + (mm * var10000) + yyyy))); newline();
    std::cin.get();
    return 0;
}
```

**Протокол запуска задачи на C++.**

**>**

```
Happy birthday to you!
        29041999 (decimal)
        11110010101001001101110110(reversed binary)
        even?   no
        odd?    yes
0
```

**Протокол запуска задачи на Лиспе.**

**>**

```
Happy birthday to you!
        29041999 (decimal)
        11110010101001001101110110(reversed binary)
        even?   no
        odd?    yes
0
```

## Контрольная задача №2.

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

**>**

```
Input gramma name>b09
Gramma:b09.txt
Source>golden-section
Source:golden-section.ss
   1|;golden-section
   2|
   3|(define var05 0.5)
   4|(define var2 2)
   5|(define var3 3)
   6|(define var4 4)
   7|(define var5 5)
   8|(define var7 7)
   9|(define var109 109)
  10|(define var110 110)
  11|
  12|(define a 0)(define b 2)
  13|(define z 0)
  14|(define (fun x)
  15| (set! x (- x (/ var109 var110 e)))
  16| (set! z  x)
  17| (- (* var5 (expt (log (expt (atan (- z var2)) var2)) var4)) z var7)
  18|;     5*ln^4(arctg^2(z-2))                        - z - 7
  19|)
  20|(define eps 0.00001)
  21|(define (golden-section-search a b)
  22| (let(
  23|      (xmin (cond ((< a b) (golden-start a b)) (#t (golden-start b a ))))
  24|     )
  25|     (newline)
  26|     xmin
  27| )
  28|)
  29|(define (golden-start a b)
  30| (set! mphi(* var05(- var3(sqrt var5))))
  31| (let(
  32|      (xa (+ a (* mphi(- b a))))
  33|      (xb (- b (* mphi(- b a))))
  34|     )
  35|     (try a b xa (fun xa) xb (fun xb))
  36| )
  37|)
  38|(define mphi 0)
  39|(define (try a b xa ya xb yb)
```

```
 40| (cond((<(abs (- a b))eps) (* (+ a b) var05))
 41|      (#t (display "+")
 42|          (cond((< ya yb)(set! b xb)
 43|                          (set! xb xa)
 44|                          (set! yb ya)
 45|                          (set! xa (+ a (* mphi(- b a))))
 46|                          (try a b xa (fun xa) xb yb)
 47|                )
 48|                (#t    (set! a xa)
 49|                          (set! xa xb)
 50|                          (set! ya yb)
 51|                          (set! xb (- b (* mphi(- b a))))
 52|                          (try a b xa ya xb (fun xb))
 53|                )
 54|          )
 55|       )
 56| )
 57|)
 58|(define xmin 0)
 59|(set! xmin(golden-section-search a b))
 60|  (display"interval=\t[")
 61|  (display a)
 62|  (display" , ")
 63|  (display b)
 64|  (display"]\n")
 65|  (display"xmin=\t\t")
 66|xmin
 67|  (display"f(xmin)=\t")
 68|(fun xmin)
 69|
```

---

Code:

```
/*  KAV2019   */
#include "mlisp.h"

double var05 = 0.5;
double var2 = 2;
double var3 = 3;
double var4 = 4;
double var5 = 5;
double var7 = 7;
double var109 = 109;
double var110 = 110;
double a = 0;
double b = 2;
```

```
double z = 0;
double eps = 0.00001;
double mphi = 0;
double xmin = 0;

double fun(double x);
double golden__section__search(double a, double b);
double golden__start(double a, double b);
double __KAV2019__try(double a, double b, double xa, double ya, double xb, double yb);
//_________________
double fun(double x){
    x = (x - (double(var109) / var110 / e));
    z = x;
    return ((var5 * expt(log(expt(atan((z - var2)), var2)), var4)) - z - var7);
}

double golden__section__search(double a, double b){
    {
        double xmin((((a < b) ? golden__start(a, b)
            : (true ? golden__start(b, a)
                : _infinity))));
        newline();
        return xmin;
    }
}

double golden__start(double a, double b){
    mphi = (var05 * (var3 - sqrt(var5)));
    {
        double xa((a + (mphi * (b - a)))),
        xb((b - (mphi * (b - a))));
        return __KAV2019__try(a, b, xa, fun(xa), xb, fun(xb));
    }
}

double __KAV2019__try(double a, double b, double xa, double ya, double xb, double yb){
    return (((abs((a - b)) < eps) ? ((a + b) * var05)
        : (true ? display("+"),
                (((ya < yb) ? b = xb,
                        xb = xa,
                        yb = ya,
                        xa = (a + (mphi * (b - a))),
                        __KAV2019__try(a, b, xa, fun(xa), xb, yb)
                    : (true ? a = xa,
                            xa = xb,
```

```
                          ya = yb,
                          xb = (b - (mphi * (b - a))),
                          __KAV2019__try(a, b, xa, ya, xb, fun(xb))
                      : _infinity)))
           : _infinity)));
}

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++.**

**>**

```
++++++++++++++++++++++++++++++
interval=       [0 , 2]
xmin=           1.258902201321729
f(xmin)=        -7.810949504291536
```

**Протокол запуска задачи на Лиспе.**

**>**

```
++++++++++++++++++++++++++
interval=       [0 , 2]
xmin=           1.2589022013217295
f(xmin)=        -7.810949504291536
```

## Контрольная задача №3.

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

**>**

```
Input gramma name>b09
Gramma:b09.txt
Source>coin19
Source:coin19.ss
   1|(define VARIANT 9)
   2|(define LAST-DIGIT-OF-GROUP-NUMBER 8)
   3|(define LARGEST-COIN 20)
   4|
   5|(define var1 1)
   6|(define var2 2)
   7|(define var3 3)
   8|(define var5 5)
   9|(define var10 10)
  10|(define var15 15)
  11|(define var20 20)
  12|(define var100 100)
  13|(define var137 137)
  14|
  15|(define (my-not? x?)
  16|  (= 0 (cond (x? e) (#t 0)) )
  17|)
  18|
  19|(define (my-or? x? y?)
  20|  (= e (cond (x? e) (#t (cond (y? e) (#t 0))) ) )
  21|)
  22|
  23|(define (implication? x? y?)
  24|  (my-or? (my-not? x?) y?)
  25|)
  26|
  27|(define (cc amount largest-coin) (cond ((my-or? (= amount 0) (= largest-coin var1)) var1
)
  28|                                ((implication? (my-not? (< amount 0)) (= largest-coin 0
)) 0)
  29|                                (#t (+ (cc amount (next-coin largest-coin))
  30|                                   (cc (- amount largest-coin) largest-coin))) )
  31|)
  32|
  33|(define (count-change amount) (cc amount LARGEST-COIN))
  34|
  35|(define (next-coin coin) (cond ((= coin var20) var15)
  36|                        ((= coin var15) var10)
  37|                        ((= coin var10) var5)
```

```
  38|                        ((= coin var5) var3)
  39|                        ((= coin var3) var2)
  40|                        (#t var1) )
  41|)
  42|
  43|(define (GR-AMOUNT) (remainder (+ (* var100 LAST-DIGIT-OF-GROUP-NUMBER) VARIANT) var137)
)
  44|
  45|(display " KAV variant ")
  46|(display VARIANT) (newline)
  47|(display " 1-2-3-5-10-15-20") (newline)
  48|(display "count__change for 100 \t= ")
  49|(display (count-change var100)) (newline)
  50|(display "count__change for ");
  51|(display (GR-AMOUNT))
  52|(display " \t= ")
  53|(display (count-change (GR-AMOUNT)))(newline)
  54|
_________________
Code:
/*  KAV2019   */
#include "mlisp.h"

double VARIANT = 9;
double LAST__DIGIT__OF__GROUP__NUMBER = 8;
double LARGEST__COIN = 20;
double var1 = 1;
double var2 = 2;
double var3 = 3;
double var5 = 5;
double var10 = 10;
double var15 = 15;
double var20 = 20;
double var100 = 100;
double var137 = 137;

bool my__not_Q(bool x_Q);
bool my__or_Q(bool x_Q, bool y_Q);
bool implication_Q(bool x_Q, bool y_Q);
double cc(double amount, double largest__coin);
double count__change(double amount);
double next__coin(double coin);
double GR__AMOUNT();
//_________________
bool my__not_Q(bool x_Q){
```

```
    return (0 == ((x_Q ? e
        : (true ? 0
            : _infinity))));
}
bool my__or_Q(bool x_Q, bool y_Q){
    return (e == ((x_Q ? e
        : (true ? ((y_Q ? e
                    : (true ? 0
                        : _infinity)))
            : _infinity))));
}
bool implication_Q(bool x_Q, bool y_Q){
    return my__or_Q(my__not_Q(x_Q), y_Q);
}
double cc(double amount, double largest__coin){
    return ((my__or_Q((amount == 0), (largest__coin == var1)) ? var1
        : (implication_Q(my__not_Q((amount < 0)), (largest__coin == 0)) ? 0
            : (true ? (cc(amount, next__coin(largest__coin)) + cc((amount - largest__coin), l
argest__coin))
                : _infinity))));
}

double count__change(double amount){
    return cc(amount, LARGEST__COIN);
}

double next__coin(double coin){
    return (((coin == var20) ? var15
        : ((coin == var15) ? var10
            : ((coin == var10) ? var5
                : ((coin == var5) ? var3
                    : ((coin == var3) ? var2
                        : (true ? var1
                            : _infinity)))))));
}

double GR__AMOUNT(){
    return remainder(((var100 * LAST__DIGIT__OF__GROUP__NUMBER) + VARIANT), var137);
}

int main(){
    display(" KAV variant ");
    display(VARIANT);
    newline();
    display(" 1-2-3-5-10-15-20");
```

```
    newline();
    display("count__change for 100 \t= ");
    display(count__change(var100));
    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++.**
**>**

```
 KAV variant 9
 1-2-3-5-10-15-20
count__change for 100   = 63992
count__change for 124   = 182492
```

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

```
 KAV variant 9
 1-2-3-5-10-15-20
count__change for 100   = 63992
count__change for 124   = 182492
```

**Распечатка файла code-gen.cpp.**
**>**

```
/* $b09 */
#include "code-gen.h"

#define TAB_SIZE 4 // количество пробелов в одном знаке табуляции

const std::string oneTab(TAB_SIZE, ' '); // чтобы не конструировать строку несколько раз
std::string globalVars; // накапливает определения глобальных переменных
int tabCount = 0;       // означает количество знаков табуляции, нужное для формирования кода с текущим уровнем вложенности

using namespace std;
int tCG::p01() {  // S -> PROG
    string header = "/*  " + lex.Authentication() + "   */\n";
    header += "#include \"mlisp.h\"\n\n";
    header += globalVars + "\n";

    // clearing before the next translation
    globalVars.clear();
    tabCount = 0;

    header += declarations;
    header += "//_________________ \n";
    S1->obj = header + S1->obj;
```

```
   return 0;
}
int tCG::p02() {  //   PROG -> CALCS
   S1->obj = "int main(){\n" + S1->obj + oneTab +
"std::cin.get();\n" + oneTab + "return 0;\n}\n";
   return 0;
}
int tCG::p03() {  //   PROG -> DEFS
   S1->obj += "int main(){\n";
   S1->obj.append(TAB_SIZE, ' ');
   S1->obj += "display(\"No calculations!\");newline();\n";
   S1->obj.append(TAB_SIZE, ' ');
   S1->obj += "std::cin.get();\n";
   S1->obj.append(TAB_SIZE, ' ');
   S1->obj += "return 0;\n}\n";
   return 0;
}
int tCG::p04() {  //   PROG -> DEFS CALCS
   S1->obj += "int main(){\n";
   S1->obj += S2->obj;
   S1->obj.append(TAB_SIZE, ' ');
   S1->obj += "std::cin.get();\n";
   S1->obj.append(TAB_SIZE, ' ');
   S1->obj += "return 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
   S1->obj = oneTab + "display(" + S1->obj + "); newline();\
n";
   return 0;
}
int tCG::p08() {  //   CALC -> BOOL
   S1->obj = oneTab + "display(" + S1->obj + "); newline();\
n";
   return 0;
}
int tCG::p09() {  //   CALC -> STR
```

```
   S1->obj = oneTab + "display(" + S1->obj + "); newline();\
n";
   return 0;
}
int tCG::p10() {  //   CALC -> DISPSET
   S1->obj = oneTab + 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
   return 0;
}
int tCG::p14() {  //      E -> SUB
   return 0;
}
int tCG::p15() {  //      E -> DIV
   return 0;
}
int tCG::p16() {  //      E -> MUL
   return 0;
}
int tCG::p17() {  //      E -> COND
   return 0;
}
int tCG::p18() {  //      E -> CPROC
   return 0;
}
int tCG::p19() {  //    ADD -> HADD E )
   S1->obj += (S1->count > 0 ? " + " : "+");
   S1->obj += S2->obj;
   S1->obj += ")";
   return 0;
}
int tCG::p20() {  //   HADD -> ( +
   S1->obj = "(";
   return 0;
}
```

```
int tCG::p21() {  //   HADD -> HADD E
   if(S1->count > 0)
      S1->obj += " + ";
   S1->obj += S2->obj;
   S1->count++;
   return 0;
}
int tCG::p22() {  //    SUB -> HSUB E )
   S1->obj += (S1->count > 0 ? " - " : "-");
   S1->obj += S2->obj;
   S1->obj += ")";
   return 0;
}
int tCG::p23() {  //   HSUB -> ( -
   S1->obj = "(";
   return 0;
}
int tCG::p24() {  //   HSUB -> HSUB E
   if (S1->count > 0)
      S1->obj += " - ";
   S1->obj += S2->obj;
   S1->count++;
   return 0;
}
int tCG::p25() {  //    DIV -> HDIV E )
   if(S1->count > 0){
      S1->obj += " / ";
   }
   S1->obj += S2->obj;
   S1->obj += ")";
   return 0;
}
int tCG::p26() {  //   HDIV -> ( /
   S1->obj = "(";
   return 0;
}
int tCG::p27() {  //   HDIV -> HDIV E
   if (S1->count > 0)
      S1->obj += " / ";
   if(S1->count == 0){
      S1->obj += "double(";
      S1->obj += S2->obj;
      S1->obj += ")";
   }
```

```
    else{
        S1->obj += S2->obj;
    }
    S1->count++;
    return 0;
}
int tCG::p28() {  //    MUL -> HMUL E )
    if (S1->count == 0)
        S1->obj = S2->obj;
    else
        S1->obj += S2->obj;
    S1->obj += ")";
    S1->count = 0;
    return 0;
}
int tCG::p29() {  //   HMUL -> ( *
    S1->obj = "(";
    return 0;
}
int tCG::p30() {  //   HMUL -> HMUL E
    S1->obj += S2->obj;
    S1->obj += " * ";
    ++S1->count;
    return 0;
}
int tCG::p31() {  //   COND -> HCOND CLAUS )
    S1->obj += S2->obj;
    S1->obj += "\n";
    S1->obj.append(tabCount, ' ');
    S1->obj += ": _infinity";
    S1->obj.append(S1->count + 1, ')');

    tabCount -= (S1->count + 1) * TAB_SIZE;

    S1->obj += ")";
    return 0;
}
int tCG::p32() {  //  HCOND -> ( cond
    S1->obj = "(";
    tabCount += TAB_SIZE;
    return 0;
}
int tCG::p33() {  //  HCOND -> HCOND CLAUS
    S1->obj += S2->obj;
```

```
    S1->obj += "\n";
    S1->obj.append(tabCount, ' ');
    S1->obj += ": ";
    S1->count++;
    tabCount += TAB_SIZE;
    return 0;
}
int tCG::p34() {  //  CLAUS -> HCLAUS E )
    if (S1->count > 0) {
        S1->obj.append(tabCount, ' ');
    }
    S1->obj += S2->obj;
    tabCount -= TAB_SIZE;
    return 0;
}
int tCG::p35() {  // HCLAUS -> ( BOOL
    S1->obj = "(" + S2->obj + " ? ";
    tabCount += TAB_SIZE;
    return 0;
}
int tCG::p36() {  // HCLAUS -> HCLAUS INTER
    if(S1->count > 0){
        S1->obj.append(tabCount, ' ');
    }
    S1->obj += S2->obj + ",\n";
    S1->count++;

    return 0;
}
int tCG::p37() {  //   ELSE -> HELSE E )
    return 0;
}
int tCG::p38() {  //  HELSE -> ( else
    return 0;
}
int tCG::p39() {  //  HELSE -> HELSE INTER
    return 0;
}
int tCG::p40() {  //  CPROC -> HCPROC )
    S1->obj += ")";
    return 0;
}
int tCG::p41() {  // HCPROC -> ( $id
    S1->obj = decor(S2->name) + "(";
```

```
    return 0;
}
int tCG::p42() {  // HCPROC -> HCPROC E
    if (S1->count > 0)
        S1->obj += ", ";
    S1->obj += S2->obj;
    S1->count++;
    return 0;
}
int tCG::p43() {  //   BOOL -> $bool
    S1->obj = (S1->name == "#t" ? "true" : "false");
    return 0;
}
int tCG::p44() {  //   BOOL -> $idq
    S1->obj = decor(S1->name);
    return 0;
}
int tCG::p45() {  //   BOOL -> CPRED
    return 0;
}
int tCG::p46() {  //   BOOL -> REL
    return 0;
}
int tCG::p47() {  //  CPRED -> HCPRED )
    S1->obj += ")";
    return 0;
}
int tCG::p48() {  // HCPRED -> ( $idq
    S1->obj = decor(S2->name) + "(";
    return 0;
}
int tCG::p49() {  // HCPRED -> HCPRED ARG
    if(S1->count > 0)
        S1->obj += ", ";
    S1->obj += S2->obj;
    S1->count++;
    return 0;
}
int tCG::p50() {  //    ARG -> E
    return 0;
}
int tCG::p51() {  //    ARG -> BOOL
    return 0;
}
```

```
int tCG::p52() {  //    REL -> ( < E E )
   S1->obj = "(" + S3->obj + " < " + S4->obj + ")";
   return 0;
}
int tCG::p53() {  //    REL -> ( = E E )
   S1->obj = "(" + S3->obj + " == " + S4->obj + ")";
   return 0;
}
int tCG::p54() {  //    STR -> $str
   S1->obj = S1->name;
   return 0;
}
int tCG::p55() {  //    STR -> SIF
   return 0;
}
int tCG::p56() {  //    SIF -> ( if BOOL STR STR )
   S1->obj = "(" + S3->obj + " ? " + S4->obj + " : " + S5->obj
+ ")";
   return 0;
}
int tCG::p57() {  //    SET -> ( set! $id E )
   S1->obj = decor(S3->name) + " = " + S4->obj;
   return 0;
}
int tCG::p58() {  //      DISPSET -> ( display E )
   S1->obj = "display(" + S3->obj + ")";
   return 0;
}
int tCG::p59() {  //      DISPSET -> ( display BOOL )
   S1->obj = "display(" + S3->obj + ")";
   return 0;
}
int tCG::p60() {  //      DISPSET -> ( display STR )
   S1->obj = "display(" + S3->obj + ")";
   return 0;
}
int tCG::p61() {  //      DISPSET -> ( newline )
   S1->obj = "newline()";
   return 0;
}
int tCG::p62() {  //      DISPSET -> SET
   return 0;
}
int tCG::p63() {  //  INTER -> DISPSET
```

```
    return 0;
}
int tCG::p64() {  //  INTER -> E
    return 0;
}
int tCG::p65() {  //   DEFS -> DEF
    return 0;
}
int tCG::p66() {  //   DEFS -> DEFS DEF
    S1->obj += S2->obj;
    return 0;
}
int tCG::p67() {  //    DEF -> PRED
    return 0;
}
int tCG::p68() {  //    DEF -> VAR
    return 0;
}
int tCG::p69() {  //    DEF -> PROC
    return 0;
}
int tCG::p70() {  //   PRED -> HPRED BOOL )
    S1->obj += S2->obj;
    S1->obj += ";\n}\n\n";
    tabCount -= TAB_SIZE;
    return 0;
}
int tCG::p71() {  //  HPRED -> PDPAR )
    S1->obj += ")";
    declarations += S1->obj + ";\n";  //!!!
    S1->obj += "{\n";
    S1->obj.append(TAB_SIZE, ' ');
    S1->obj += "return ";
    S1->count = 0;
    return 0;
}
int tCG::p72() {  //  PDPAR -> ( define ( $idq
    S1->obj = "bool " + decor(S4->name) + "(";
    S1->count = 0;
    tabCount += TAB_SIZE;
    return 0;
}
int tCG::p73() {  //  PDPAR -> PDPAR $idq
    if (S1->count)
```

```
      S1->obj += ", ";
   S1->obj += "bool ";
   S1->obj += decor(S2->name);
   ++(S1->count);
   return 0;
}
int tCG::p74() {  //  PDPAR -> PDPAR $id
   if (S1->count)
      S1->obj += ", ";
   S1->obj += "double ";
   S1->obj += decor(S2->name);
   ++(S1->count);
   return 0;
}
int tCG::p75() {  //  CONST -> $zero
   S1->obj = "0";
   return 0;
}
int tCG::p76() {  //  CONST -> $dec
   S1->obj = decor(S1->name);
   return 0;
}
int tCG::p77() {  //    VAR -> ( define $id CONST )
   globalVars += "double " + decor(S3->name) + " = " + S4-
>obj + ";\n";
   S1->obj = "";
   return 0;
}
int tCG::p78() {  //   PROC -> HPROC LET )
   S1->obj += S2->obj;
   S1->obj += "}\n\n";
   tabCount -= TAB_SIZE;
   return 0;
}
int tCG::p79() {  //   PROC -> HPROC E )
   S1->obj.append(TAB_SIZE, ' ');
   S1->obj += "return ";
   S1->obj += S2->obj;
   S1->obj += ";\n}\n\n";
   tabCount -= TAB_SIZE;
   return 0;
}
int tCG::p80() {  //  HPROC -> PCPAR )
   S1->obj += ")";
```

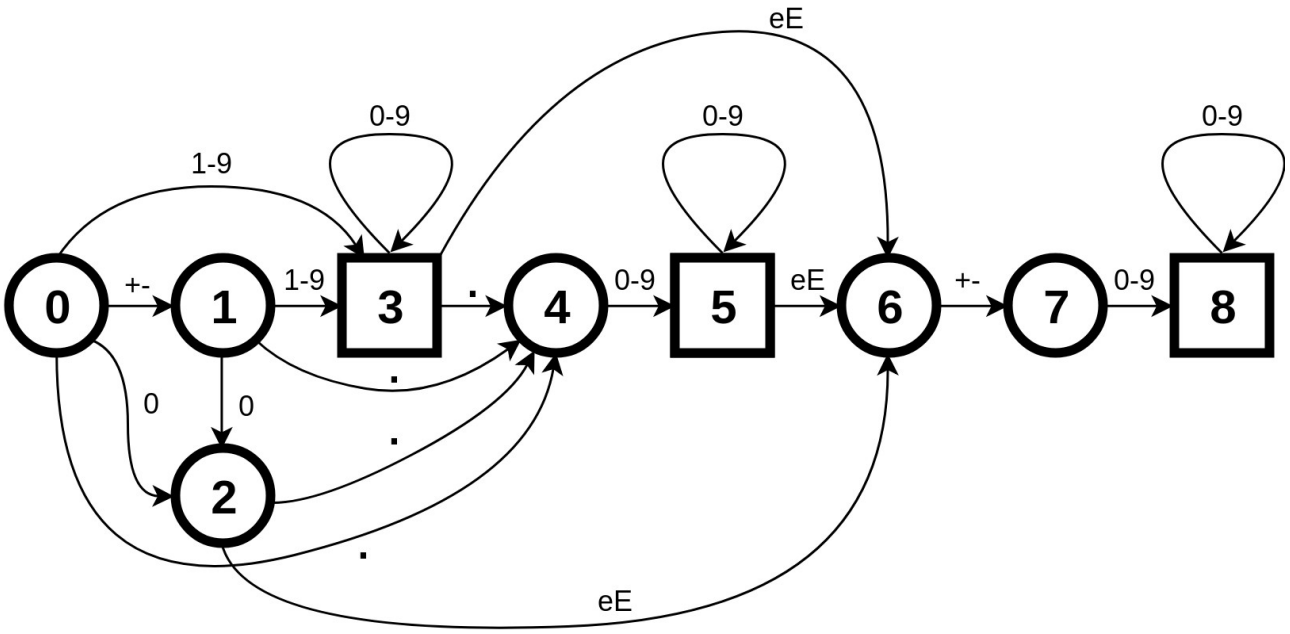
```
    declarations += S1->obj + ";\n";
    S1->obj += "{\n";

    tabCount += TAB_SIZE;

    return 0;
}
int tCG::p81() {  //  HPROC -> HPROC INTER
    S1->obj.append(tabCount, ' ');
    S1->obj += S2->obj;
    S1->obj += ";\n";
    return 0;
}
int tCG::p82() {  //  PCPAR -> ( define ( $id
    S1->obj = "double " + decor(S4->name) + "(";
    return 0;
}
int tCG::p83() {  //  PCPAR -> PCPAR $id
    if (S1->count > 0)
        S1->obj += ", ";
    S1->obj += "double ";
    S1->obj += decor(S2->name);
    S1->count++;
    return 0;
}
int tCG::p84() {  //    LET -> HLET E )
    S1->obj.append(tabCount, ' ');
    S1->obj += "return ";
    S1->obj += S2->obj;
    S1->obj += ";\n";
    S1->obj.append(TAB_SIZE, ' ');
    S1->obj += "}\n";
    tabCount -= TAB_SIZE;
    return 0;
}
int tCG::p85() {  //   HLET -> LETLOC )
    S1->obj = oneTab + "{\n" + S1->obj + ";\n";
    return 0;
}
int tCG::p86() {  //   HLET -> HLET INTER
    S1->obj.append(tabCount, ' ');
    S1->obj += S2->obj;
    S1->obj += ";\n";
    return 0;
```

```
}
int tCG::p87() {  // LETLOC -> ( let (
   tabCount += TAB_SIZE;
   S1->obj = std::string(tabCount, ' ') + "double " + S1->obj;
   return 0;
}
int tCG::p88() {  // LETLOC -> LETLOC LETVAR
   if(S1->count > 0){
      S1->obj += ",\n";
      S1->obj.append(tabCount, ' ');
   }
   S1->obj += S2->obj;
   S1->count++;
   return 0;
}
int tCG::p89() {  // LETVAR -> ( $id E )
   S1->obj = decor(S2->name) + "(" + S3->obj + ")";
   return 0;
}
//____________________
int tCG::p90(){return 0;} int tCG::p91(){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;}
```

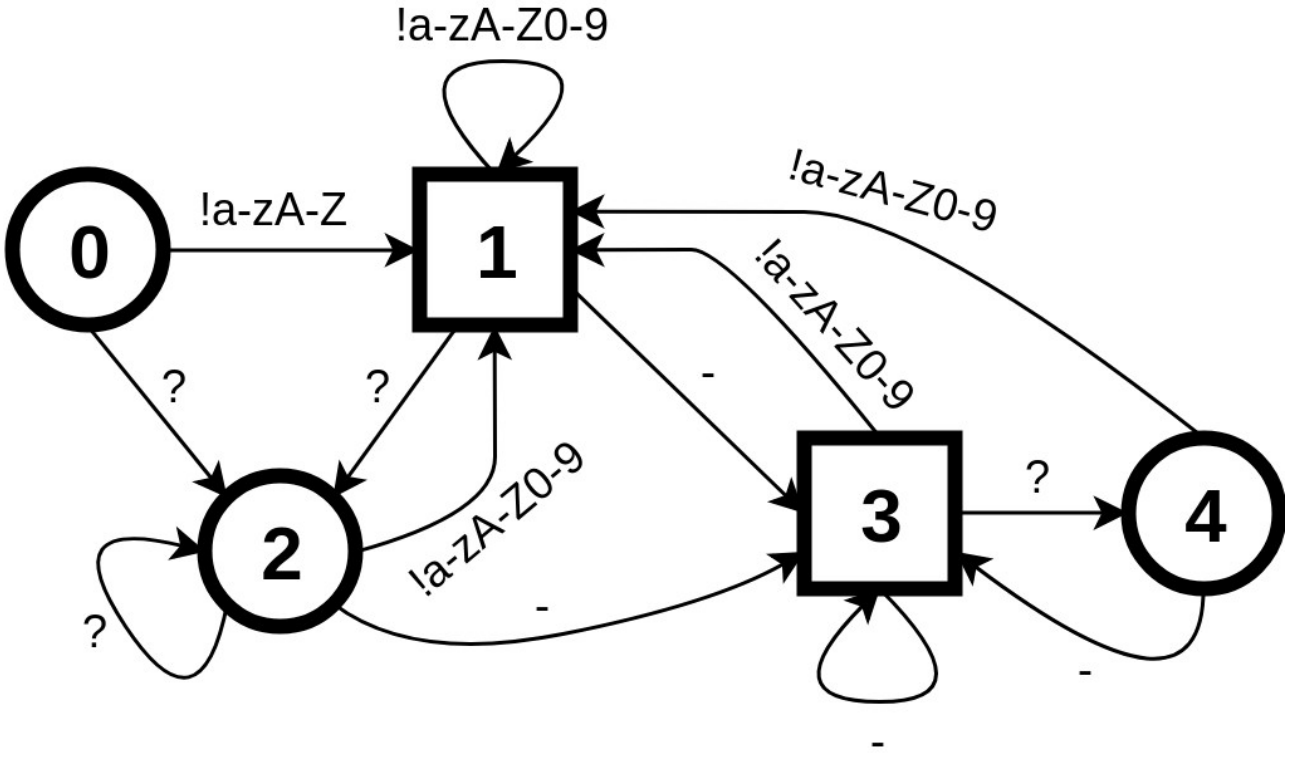
**Диаграммы автоматов из лабораторной работы №5 для токенов $dec, $id, $idq.  Над каждой диаграммой проставить  номер варианта шаблона токена и его краткое описание. Все диаграммы должны быть построены в одном редакторе и должны иметь единый стиль изображения.**

**>**

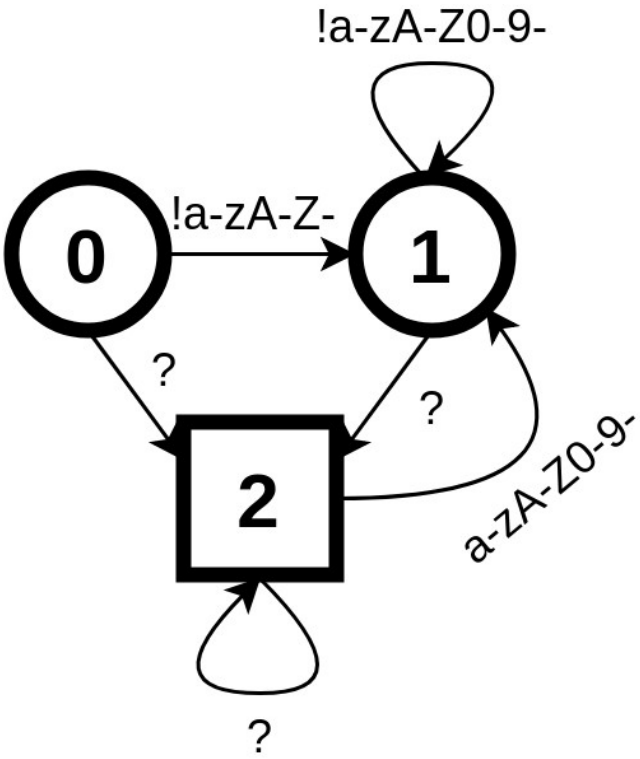
**$dec: 208-09. Целую часть можно опустить, СОХРАНЯЯ точку и дробную часть, например, .5, -.5, +.5, .5e+0 .**




**$id: 208-09. Комбинация символов -?? запрещена.**



**$idq: 208-09. Комбинация символов ?! Запрещена.**




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

**Построение синтаксического транслятора сначала показалось мне довольно трудной задачей. Пришлось даже немного почитать про восходящий анализ и LR(k) грамматики. Потом, поняв суть действий переноса-свертки и найдя эквиваленты им в коде , пришло понимание и все пошло как по маслу.**

**Из-за ограничений, наложенных конкретным вариантом грамматики возникли некоторые сложности с реализацией замены для логических операций.**

**Вероятно, используя полную грамматику m19, реализовать синтаксический транслятор было бы даже легче, несмотря на чуть больший объем работы.**

**Так же, если не заморачиваться насчет переноса строк и табуляций, код на целевом языке становится абсолютно нечитаемым. Поэтому пришлось немного усложнить себе задачу и сделать генерацию кода с табуляциями, учитывающими вложенность.**

**В наше время (как мне думалось) компиляторы уже написаны и ничего принципиально нового появляться не должно.**

**И как выяснилось сфера применений синтаксических/ лексических анализаторов не ограничивается компиляторами. Они могут применяться во всевозможных парсерах и даже для анализа естественных языков.**

**Для этого существуют специальные утилиты, делающие всю рутинную работу за нас.**

**Комбинация lex\yacc позволяет делать грубо говоря (как минимум) то же самое, что реализовано в наборе классов, данных нам уже готовыми.**
**Lex предназначен для реализации лексического анализа, yac — синтаксического анализа.**

**Почитав  для общего развития про эти утилиты я понял, что задачи, так или иначе связанные с лексическим/синтаксическим разбором, не то чтобы очень часто, но возникают на практике. Поэтому было бы неплохо знать как это устроено, и знания, полученные на протяжении курса, решают этот вопрос.**

**Как результат получен рабочий транслятор с диалекта языка МИКРОЛИСП на язык C++. При запуске программ на исходном и целевом языках их выводы совпадают. Таким образом задание первой части курсовой работы выполнено в полном объеме.**