**Студент: Поскряков Я. С.**

**Группа:М8О-206Б-17**

**Номер по списку:19**

**«СИСТЕМЫ ПРОГРАММИРОВАНИЯ»**

**Курсовая работа 2019.**

**Часть 1.**

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

**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.**

1. **Поиск минимума функции методом золотого сечения.**
2. **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.**

**># $a19**

**$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 |**

**CONST #12 |**

**ADD #13 |**

**SUB #14 |**

**DIV #15 |**

**MUL #16 |**

**COND #17 |**

**CPROC #18**

**CONST -> $zero #19 |**

**$dec #20**

**ADD -> HADD E ) #21**

**HADD -> ( + #22 |**

**HADD E #23**

**SUB -> ( - E E ) #24**

**DIV -> ( / E ) #25**

**MUL -> HMUL E ) #26**

**HMUL -> ( \* #27 |**

**HMUL E #28**

**COND -> ( cond CLAUS CLAUS ) #29**

**CLAUS -> HCLAUS E ) #30**

**HCLAUS -> ( BOOL #31 |**

**HCLAUS INTER #32**

**ELSE -> HELSE E ) #33**

**HELSE -> ( else #34 |**

**HELSE INTER #35**

**CPROC -> HCPROC ) #36**

**HCPROC -> ( $id #37 |**

**HCPROC E #38**

**BOOL -> $bool #39 |**

**$idq #40 |**

**CPRED #41 |**

**REL #42 |**

**AND #43 |**

**( not BOOL ) #44**

**CPRED -> HCPRED ) #45**

**HCPRED -> ( $idq #46 |**

**HCPRED ARG #47**

**ARG -> E #48 |**

**BOOL #49**

**REL -> ( <= E E ) #50 |**

**( = E E ) #51**

**AND -> HAND BOOL ) #52**

**HAND -> ( and #53 |**

**HAND BOOL #54**

**STR -> $str #55 |**

**SIF #56**

**SIF -> ( if BOOL STR STR ) #57**

**SET -> ( set! $id E ) #58**

**DISPSET -> ( display E ) #59 |**

**( display BOOL ) #60 |**

**( display STR ) #61 |**

**( newline ) #62 |**

**SET #63**

**INTER -> DISPSET #64 |**

**E #65**

**DEFS -> DEF #66 |**

**DEFS DEF #67**

**DEF -> PRED #68 |**

**VAR #69 |**

**PROC #70**

**PRED -> HPRED BOOL ) #71**

**HPRED -> PDPAR ) #72**

**PDPAR -> ( define ( $idq #73 |**

**PDPAR $idq #74 |**

**PDPAR $id #75**

**VAR -> ( define $id CONST ) #76**

**PROC -> HPROC LET ) #77 |**

**HPROC E ) #78**

**HPROC -> PCPAR ) #79 |**

**HPROC INTER #80**

**PCPAR -> ( define ( $id #81 |**

**PCPAR $id #82**

**LET -> HLET E ) #83**

**HLET -> LETLOC ) #84 |**

**HLET INTER #85**

**LETLOC -> ( let ( #86 |**

**LETLOC LETVAR #87**

**LETVAR -> ( $id E ) #88**

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

**>1. Вычитание.**

**\*1.2 Только два операнда.**

**(- x y)**

**2. Деление.**

**\*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)**

**5. Форма or.**

**\*5.2 Отсутствует.**

**6. Форма and.**

**\*6.1 Один и более операндов.**

**(and #t #f #f)**

**7. Форма not.**

**\*7.1 Есть.**

**(not #t )**

**8. Оператор = .**

**\*8.1 Есть.**

**(= x y)**

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

**\*9.2 (<= x y)**

**10. Форма IF для чисел.**

**\*10.2 Отсутствует.**

**11. Форма IF для строк.**

**\*11.1 Есть.**

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

**12. Форма COND.**

**\*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.**

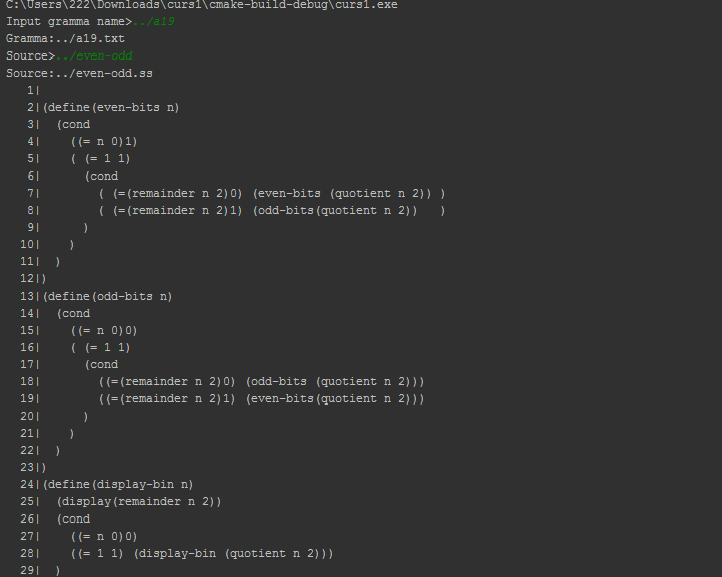
**13. Локальные переменные.**

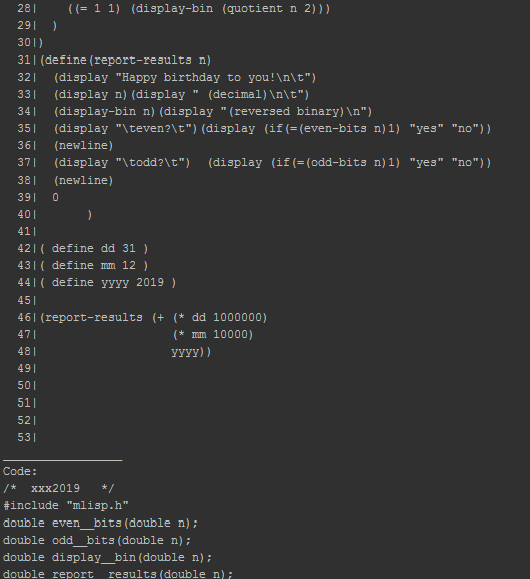
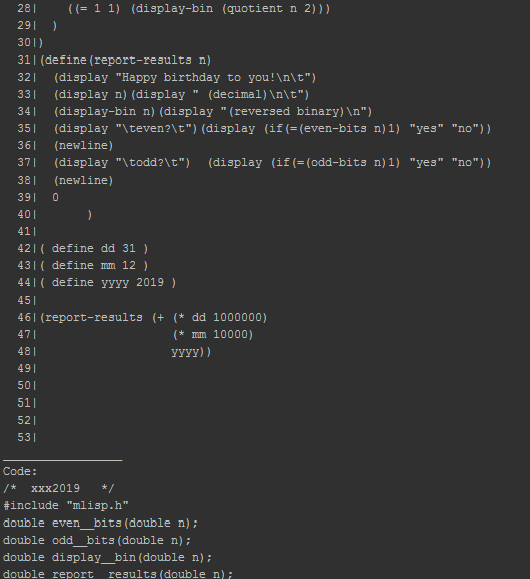
**\*13.1 Определяются формой let.**

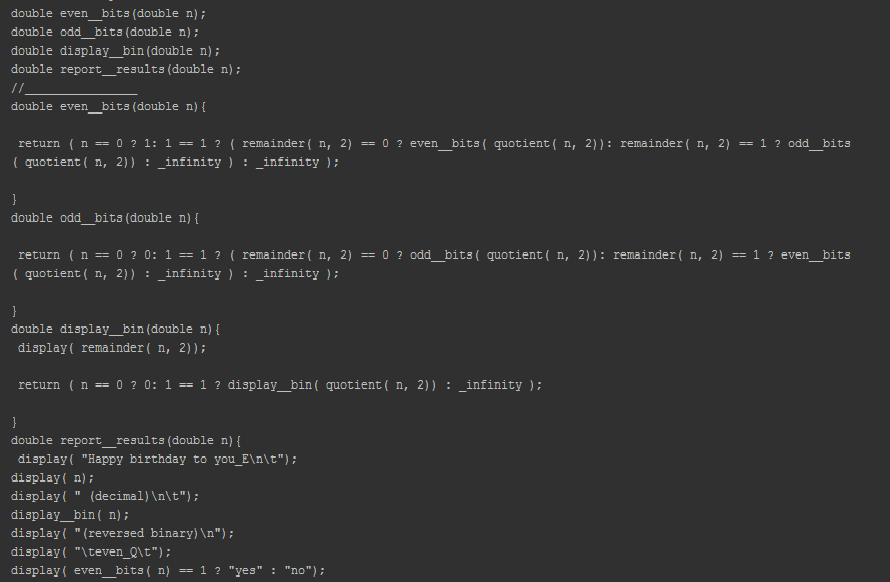
**(define (f)(let((a pi))a)) (f)**

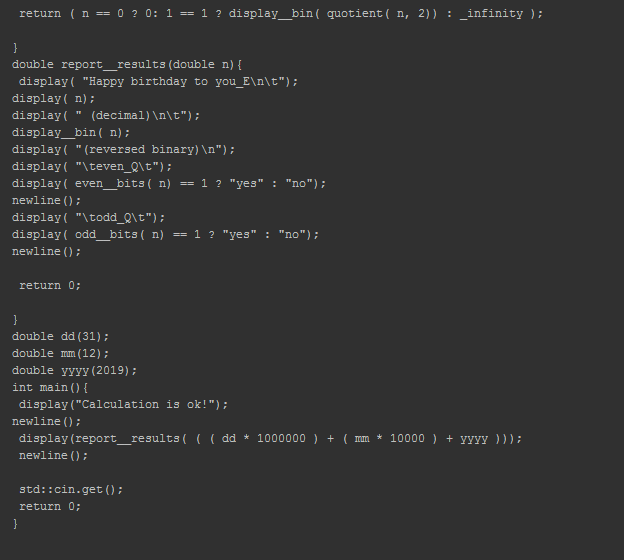
**Контрольная задача №1**

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

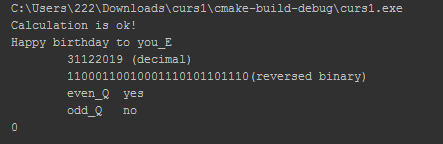
**>**



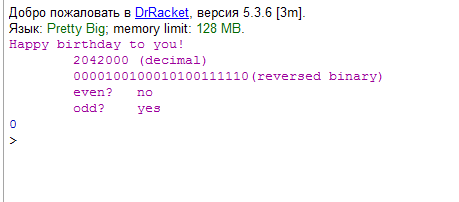




**Протокол запуска задачи на С++.**

**>**

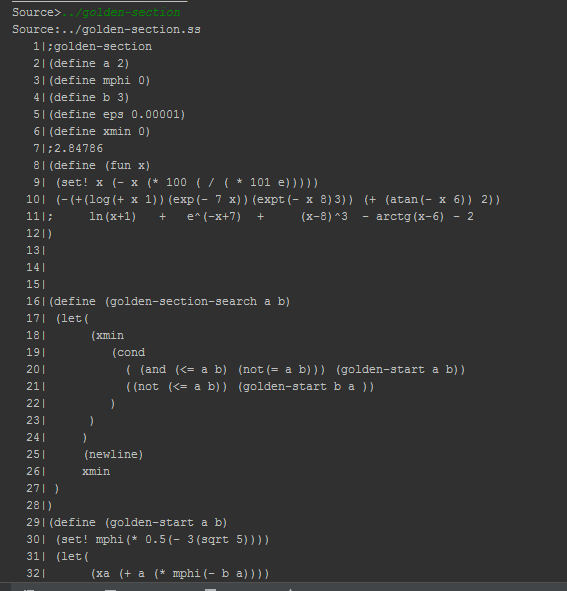
**Протокол запуска задачи на Лиспе.**

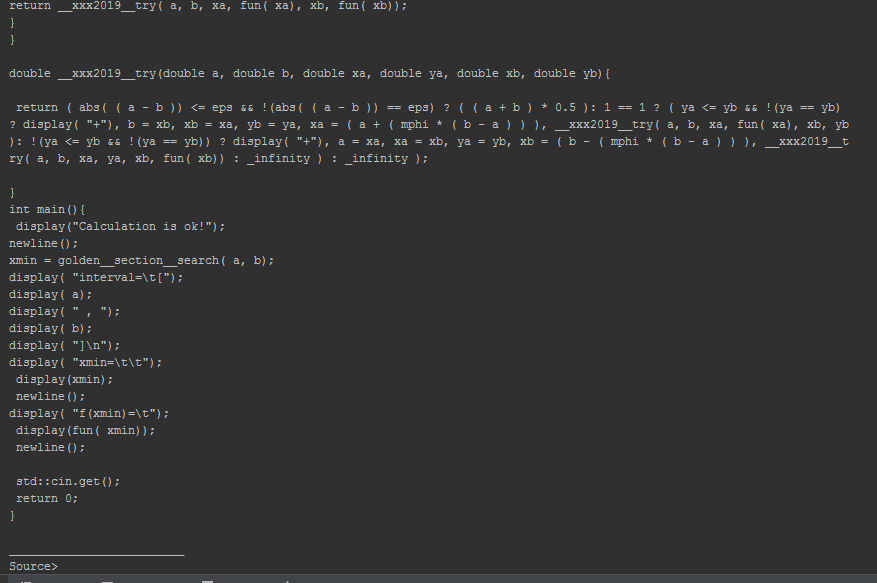
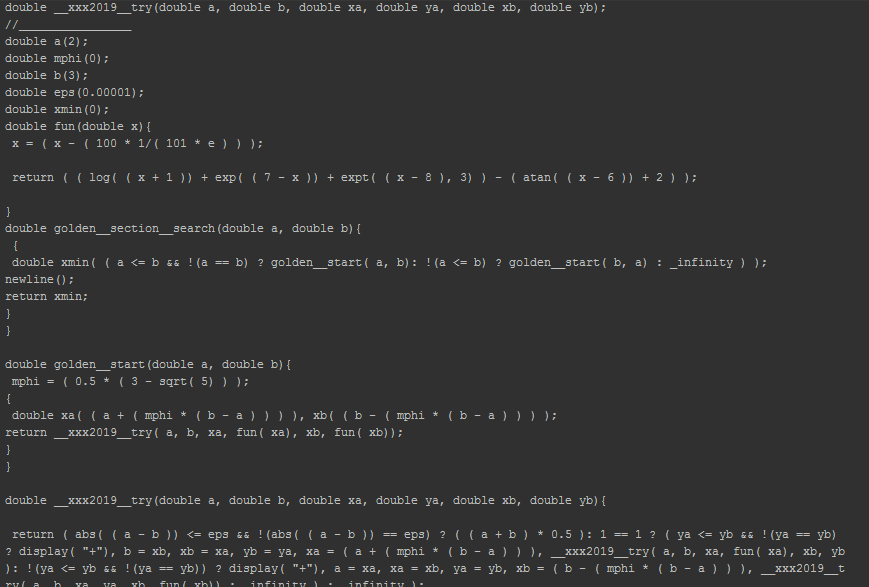
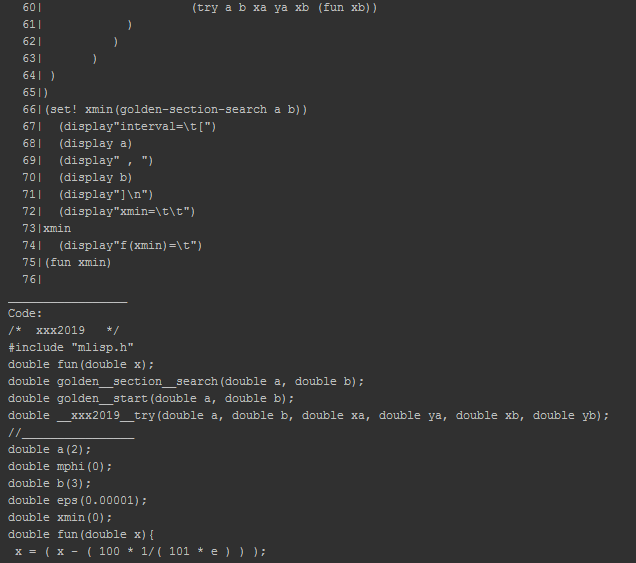
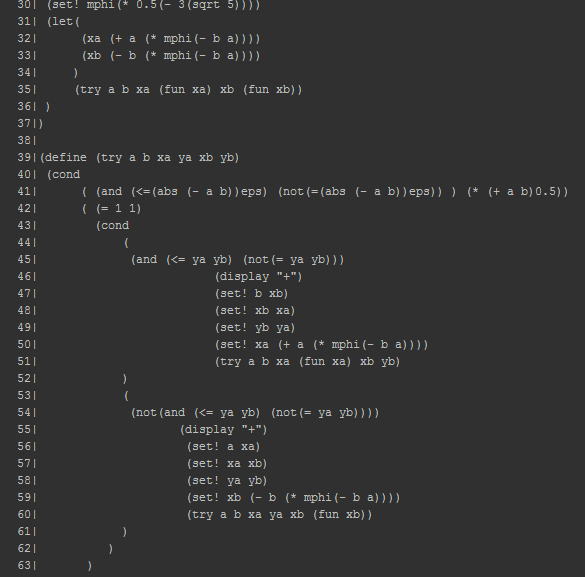
**>**

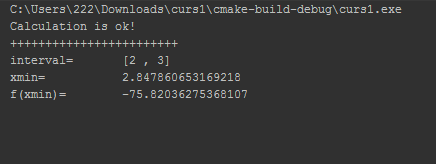
**Контрольная задача №2.**

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

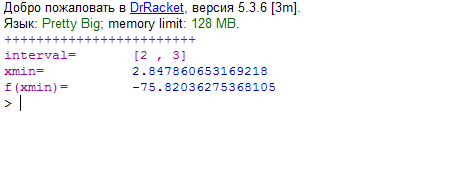
**>**



**Протокол запуска задачи на С++.**

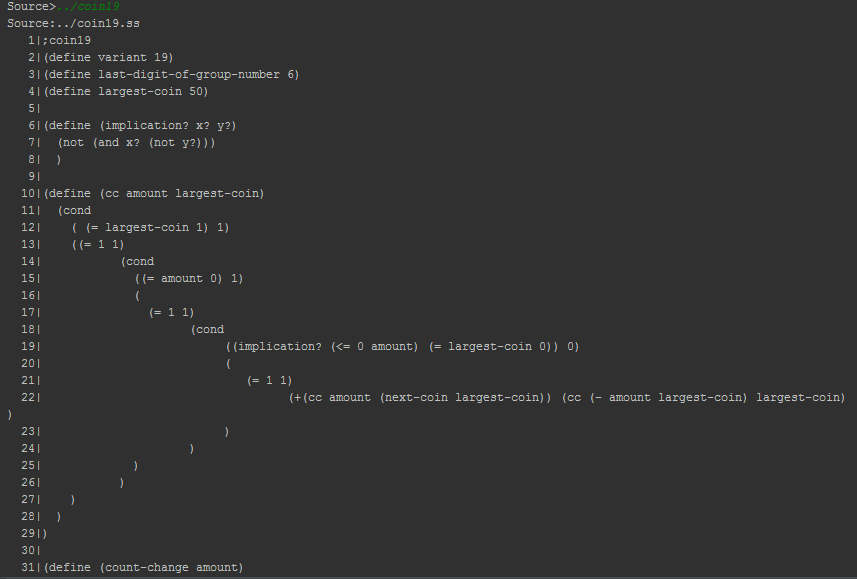
**>**

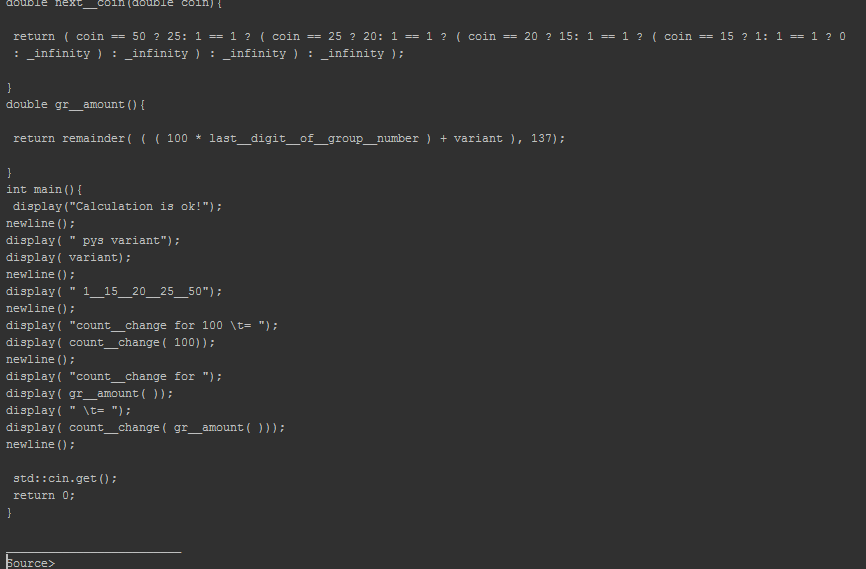
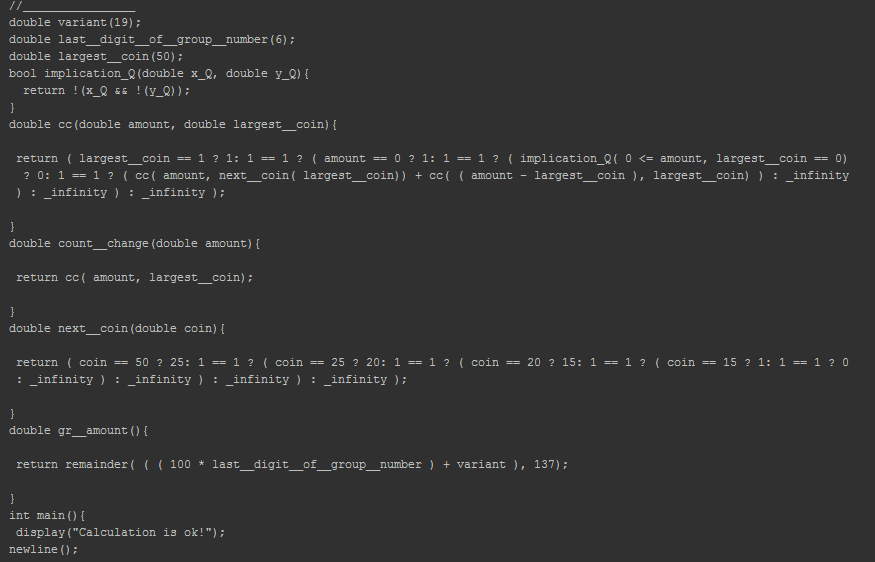
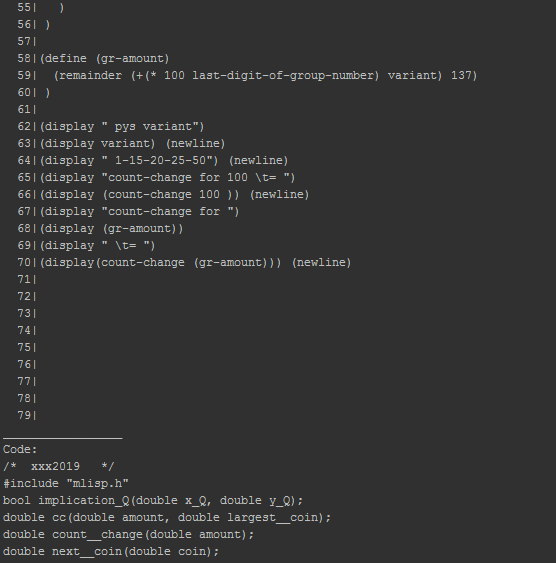
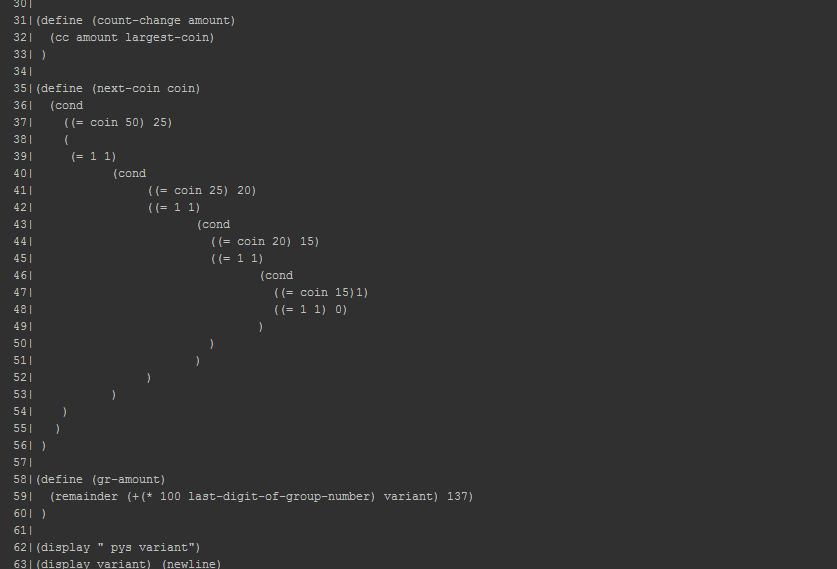
**Протокол запуска задачи на Лиспе.**

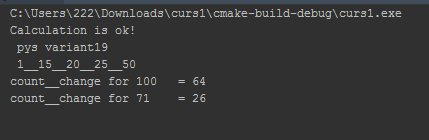
**>**

**Контрольная задача №3.**

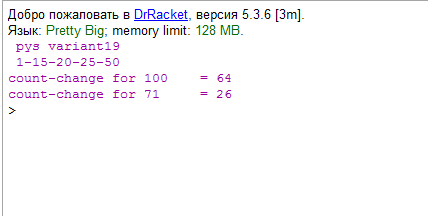
**Полный протокол трансляции без трассировки (крупный белый шрифт на ярком черном фоне).**

**>**

**Протокол запуска задачи на С++.**

**>**

**Протокол запуска задачи на Лиспе.**

**>**

**Распечатка файла code-gen.cpp.**

**>**

/\* $a19 \*/  
#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"  
 " display(\"No calculations!\");\nnewline();\n"  
 " std::cin.get();\n return 0;\n}\n";  
 return 0;  
}  
  
int tCG::p04() { // PROG -> DEFS CALCS  
 S1->obj += "int main(){\n"  
 " display(\"Calculation is ok!\");\nnewline();\n";  
 S1->obj += S2->obj;  
 S1->obj += "\n std::cin.get();\n 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 = " display(" + S1->obj + ");\n newline();\n";  
 return 0;  
}  
  
int tCG::p08() { // CALC -> BOOL  
 return 0;  
}  
  
int tCG::p09() { // CALC -> STR  
 return 0;  
}  
  
int tCG::p10() { // CALC -> DISPSET  
 S1->obj += ";\n";  
 return 0;  
}  
  
int tCG::p11() { // E -> $id  
 S1->obj = decor(S1->name);  
 return 0;  
}  
  
int tCG::p12() { // E -> CONST  
 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() { // CONST -> $zero  
 S1->obj = decor(S1->name);  
 return 0;  
}  
  
int tCG::p20() { // CONST -> $dec  
 S1->obj = decor(S1->name);  
 return 0;  
}  
  
int tCG::p21() { // ADD -> HADD E )  
 if (S1->count == 0)  
 S1->obj = S2->obj;  
 else  
 S1->obj += S2->obj;  
 S1->count = 0;  
 S1->obj += " )";  
 return 0;  
}  
  
int tCG::p22() { // HADD -> ( +  
 S1->obj += "( ";  
 return 0;  
}  
  
int tCG::p23() { // HADD -> HADD E  
 S1->obj += S2->obj + " + ";  
 ++S1->count;  
 return 0;  
}  
  
int tCG::p24() { // SUB -> ( - E E )  
 S1->obj = "( " + S3->obj + " - " + S4->obj + " )";  
 return 0;  
}  
  
int tCG::p25() { // DIV -> ( / E )  
 S1->obj = "1/" + S3->obj;  
 return 0;  
}  
  
int tCG::p26() { // MUL -> HMUL E )  
 if (S1->count == 0)  
 S1->obj = S2->obj;  
 else  
 S1->obj += S2->obj;  
 S1->count = 0;  
 S1->obj += " )";  
 return 0;  
}  
  
int tCG::p27() { // HMUL -> ( \*  
 S1->obj += "( ";  
 return 0;  
}  
  
int tCG::p28() { // HMUL -> HMUL E  
 S1->obj += S2->obj + " \* ";  
 ++S1->count;  
 return 0;  
}  
  
int tCG::p29() { // COND -> ( cond CLAUS CLAUS )  
 S1->obj += "( " + S3->obj + ": " + S4->obj + " : \_infinity )";  
 return 0;  
}  
  
int tCG::p30() { // CLAUS -> HCLAUS E )  
 if (S1->count)S1->obj += ", ";// не первый параметр  
 S1->obj += S2->obj;  
 ++(S1->count);  
 return 0;  
}  
  
int tCG::p31() { // HCLAUS -> ( BOOL  
 S1->obj += S2->obj + " ? ";  
 return 0;  
}  
  
int tCG::p32() { // HCLAUS -> HCLAUS INTER  
 if (S1->count)S1->obj += ", ";// не первый параметр  
 S1->obj += S2->obj;  
 ++(S1->count);  
 return 0;  
}  
  
int tCG::p33() { // ELSE -> HELSE E )  
 return 0;  
}  
  
int tCG::p34() { // HELSE -> ( else  
 return 0;  
}  
  
int tCG::p35() { // HELSE -> HELSE INTER  
 return 0;  
}  
  
int tCG::p36() { // CPROC -> HCPROC )  
 S1->obj += ")";  
 return 0;  
}  
  
int tCG::p37() { // HCPROC -> ( $id  
 S1->obj += decor(S2->name) + "( ";  
 return 0;  
}  
  
int tCG::p38() { // HCPROC -> HCPROC E  
 if (S1->count)S1->obj += ", ";// не первый параметр  
 S1->obj += S2->obj;  
 ++(S1->count);  
 return 0;  
}  
  
int tCG::p39() { // BOOL -> $bool  
 S1->obj = (S1->name == "#t" ? "true" : "false");  
 return 0;  
}  
  
int tCG::p40() { // BOOL -> $idq  
 S1->obj = decor(S1->name);  
 return 0;  
}  
  
int tCG::p41() { // BOOL -> CPRED  
 S1->obj = S1->obj;  
 return 0;  
}  
  
int tCG::p42() { // BOOL -> REL  
  
 return 0;  
}  
  
int tCG::p43() { // BOOL -> AND  
 return 0;  
}  
  
int tCG::p44() { // BOOL -> ( not BOOL )  
 S1->obj = "!(" + S3->obj + ")";  
 return 0;  
}  
  
int tCG::p45() { // CPRED -> HCPRED )  
 S1->obj += ") ";  
 return 0;  
}  
  
int tCG::p46() { // HCPRED -> ( $idq  
 S1->obj += decor(S2->name) + "( ";  
 return 0;  
}  
  
int tCG::p47() { // HCPRED -> HCPRED ARG  
 if (S1->count)S1->obj += ", ";// не первый параметр  
 S1->obj += S2->obj;  
 ++(S1->count);  
 return 0;  
}  
  
int tCG::p48() { // ARG -> E  
 return 0;  
}  
  
int tCG::p49() { // ARG -> BOOL  
 return 0;  
}  
  
int tCG::p50() { // REL -> ( <= E E )  
 S1->obj = S3->obj + " <= " + S4->obj;  
 return 0;  
}  
  
int tCG::p51() { // REL -> ( = E E )  
 S1->obj = S3->obj + " == " + S4->obj ;  
 return 0;  
}  
  
int tCG::p52() { // AND -> HAND BOOL )  
 S1->obj += " && " + S2->obj;  
 return 0;  
}  
  
int tCG::p53() { // HAND -> ( and  
 return 0;  
}  
  
int tCG::p54() { // HAND -> HAND BOOL  
 if (S1->count)S1->obj += " && ";// не первый параметр  
 S1->obj += S2->obj;  
 ++(S1->count);  
 return 0;  
}  
  
int tCG::p55() { // STR -> $str  
 S1->obj = decor(S1->name);  
 return 0;  
}  
  
int tCG::p56() { // STR -> SIF  
 return 0;  
}  
  
int tCG::p57() { // SIF -> ( if BOOL STR STR )  
 S1->obj = S3->obj + " ? " + S4->obj + " : " + S5->obj;  
 return 0;  
}  
  
int tCG::p58() { // SET -> ( set! $id E )  
 S1->obj = decor(S3->name) + " = " + S4->obj;  
 return 0;  
}  
  
int tCG::p59() { //DISPSET -> ( display E )  
 S1->obj += "display( " + S3->obj + ")";  
 return 0;  
}  
  
int tCG::p60() { //DISPSET -> ( display BOOL )  
 S1->obj += "display( " + S3->obj + ")";  
 return 0;  
}  
  
int tCG::p61() { //DISPSET -> ( display STR )  
 S1->obj += "display( " + S3->obj + ")";  
 return 0;  
}  
  
int tCG::p62() { //DISPSET -> ( newline )  
 S1->obj = "newline()";  
 return 0;  
}  
  
int tCG::p63() { //DISPSET -> SET  
 return 0;  
}  
  
int tCG::p64() { // INTER -> DISPSET  
  
 return 0;  
}  
  
int tCG::p65() { // INTER -> E  
 return 0;  
}  
  
int tCG::p66() { // DEFS -> DEF  
 return 0;  
}  
  
int tCG::p67() { // DEFS -> DEFS DEF  
 S1->obj += S2->obj;  
 return 0;  
}  
  
int tCG::p68() { // DEF -> PRED  
 return 0;  
}  
  
int tCG::p69() { // DEF -> VAR  
 return 0;  
}  
  
int tCG::p70() { // DEF -> PROC  
 return 0;  
}  
  
int tCG::p71() { // PRED -> HPRED BOOL )  
 S1->obj += S2->obj + ";\n}\n";  
 return 0;  
}  
  
int tCG::p72() { // HPRED -> PDPAR )  
 S1->obj += ")";  
 declarations += S1->obj + ";\n"; //!!!  
 S1->obj += "{\n return ";  
 S1->count = 0;  
 return 0;  
}  
  
int tCG::p73() { // PDPAR -> ( define ( $idq  
 S1->obj += "bool " + decor(S4->name) + "(";  
 return 0;  
}  
  
int tCG::p74() { // PDPAR -> PDPAR $idq  
 if (S1->count)S1->obj += ", ";// не первый параметр  
 S1->obj += "double " + decor(S2->name);  
 ++(S1->count);  
 return 0;  
}  
  
int tCG::p75() { // PDPAR -> PDPAR $id  
 if (S1->count)S1->obj += ", ";// не первый параметр  
 S1->obj += "double " + decor(S2->name);  
 ++(S1->count);  
 return 0;  
}  
  
int tCG::p76() { // VAR -> ( define $id CONST )  
 S1->obj += "double " + decor(S3->name) + "(" + S4->obj + ");\n";  
 return 0;  
}  
  
int tCG::p77() { // PROC -> HPROC LET )  
 S1->obj += S2->obj + "\n}" + "\n\n";  
 return 0;  
}  
  
int tCG::p78() { // PROC -> HPROC E )  
 S1->obj += "\n return " + S2->obj + ";\n\n}" + "\n";  
 return 0;  
}  
  
int tCG::p79() { // HPROC -> PCPAR )  
 S1->obj += ")";  
 declarations += S1->obj + ";\n"; //!!!  
 S1->obj += "{\n "; //S1->obj += "{\n return ";  
 S1->count = 0;  
 return 0;  
}  
  
int tCG::p80() { // HPROC -> HPROC INTER  
 S1->obj += S2->obj + ";\n";  
 return 0;  
}  
  
int tCG::p81() { // PCPAR -> ( define ( $id  
 S1->obj = "double " + decor(S4->name) + "(";  
 S1->count = 0;  
 return 0;  
}  
  
int tCG::p82() { // PCPAR -> PCPAR $id  
 if (S1->count)S1->obj += ", ";// не первый параметр  
 S1->obj += "double " + decor(S2->name);  
 ++(S1->count);  
 return 0;  
}  
  
int tCG::p83() { // LET -> HLET E )  
 S1->obj += "return " + S2->obj + ";\n}";  
 return 0;  
}  
  
int tCG::p84() { // HLET -> LETLOC )  
 S1->obj += ";\n";  
 return 0;  
}  
  
int tCG::p85() { // HLET -> HLET INTER  
 S1->obj += S2->obj + ";\n";  
 return 0;  
}  
  
int tCG::p86() { // LETLOC -> ( let (  
 S1->obj += "{\n double ";  
 return 0;  
}  
  
int tCG::p87() { // LETLOC -> LETLOC LETVAR  
 if (S1->count)S1->obj += ", ";// не первый параметр  
 S1->obj += S2->obj;  
 ++(S1->count);  
 return 0;  
}  
  
int tCG::p88() { // LETVAR -> ( $id E )  
 S1->obj += decor(S2->name) + "( " + S3->obj + " )";  
 return 0;  
}  
  
//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
int tCG::p89() { 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; }