**Студент:Габдуллин Т.А.**

**Группа: М80-206-Б**

**Номер по списку: 20**

**«СИСТЕМЫ ПРОГРАММИРОВАНИЯ» Курсовой проект 2018. 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.**

**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.** |  |

**# $e20**

**$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 |**

**IF #19 |**

**CPROC #20**

**ADD -> HADD E1 ) #21**

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

**HADD E1 #23**

**SUB -> HSUB E1 ) #24**

**HSUB -> ( - E #25**

**DIV -> HDIV E1 ) #26**

**HDIV -> ( / E #27**

**MUL -> HMUL E1 ) #28**

**HMUL -> ( \* #29 |**

**HMUL E1 #30**

**IF -> IFTRUE E1 ) #31**

**IFTRUE -> HIF E #32**

**HIF -> ( if BOOL #33**

**CPROC -> HCPROC ) #34**

**HCPROC -> ( $id #35 |**

**HCPROC E #36**

**BOOL -> $bool #37 |**

**CPRED #38 |**

**REL #39 |**

**OR #40 |**

**( not BOOL ) #41**

**CPRED -> HCPRED ) #42**

**HCPRED -> ( $idq #43 |**

**HCPRED E #44**

**REL -> HREL E1 ) #45**

**HREL -> ( <= E #46 |**

**( = E #47**

**OR -> HOR BOOL ) #48**

**HOR -> ( or #49 |**

**HOR BOOL #50**

**STR -> $str #51 |**

**SIF #52**

**SIF -> SIFTRUE STR ) #53**

**SIFTRUE -> HIF STR #54**

**SET -> HSET E1 ) #55**

**HSET -> ( set! $id #56**

**DISPSET -> ( display E1 ) #57 |**

**( display BOOL ) #58 |**

**( display STR ) #59 |**

**( newline ) #60 |**

**SET #61**

**DEFS -> DEF #62 |**

**DEFS DEF #63**

**DEF -> PRED #64 |**

**PROC #65**

**PRED -> HPRED BOOL ) #66**

**HPRED -> PDPAR ) #67**

**PDPAR -> ( define ( $idq #68 |**

**PDPAR $id #69**

**PROC -> HPROC $dec ) #70 |**

**HPROC E1 ) #71**

**HPROC -> PCPAR ) #72 |**

**HPROC DISPSET #73**

**PCPAR -> ( define ( $id #74 |**

**PCPAR $id #75**

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

**Особенности диалектов языка МИКРОЛИСП 2018.**

**1. Вычитание.**

**\*1.2 Только два операнда.**

**(- x y)**

**2. Деление.**

**\* 2.2 Только два операнда.**

**(/ x y)**

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

**\*03.1 В общем контексте числового выражения.**

**4. Числовые литералы токена $dec.**

**\* 4.3 Только в определении процедуры.**

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

**5. Форма OR.**

**\* 5.1 Один и более операндов.**

**(or #t #f #f)**

**6. Форма AND.**

**\* 6.2 Отсутствует.**

**7. Оператор = .**

**\* 7.1 Есть.**

**(= x y)**

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

**\*8.2 (<= x y)**

**9. Форма IF для чисел.**

**\*9.1 Есть.**

**(if #t e pi)**

**10. Форма IF для строк.**

**\* 10.1 Есть.**

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

**11. Форма COND.**

**\*11.5 Отсутствует.**

**12. Глобальные переменные.**

**\*12.1 Есть**

**(define a 1)a**

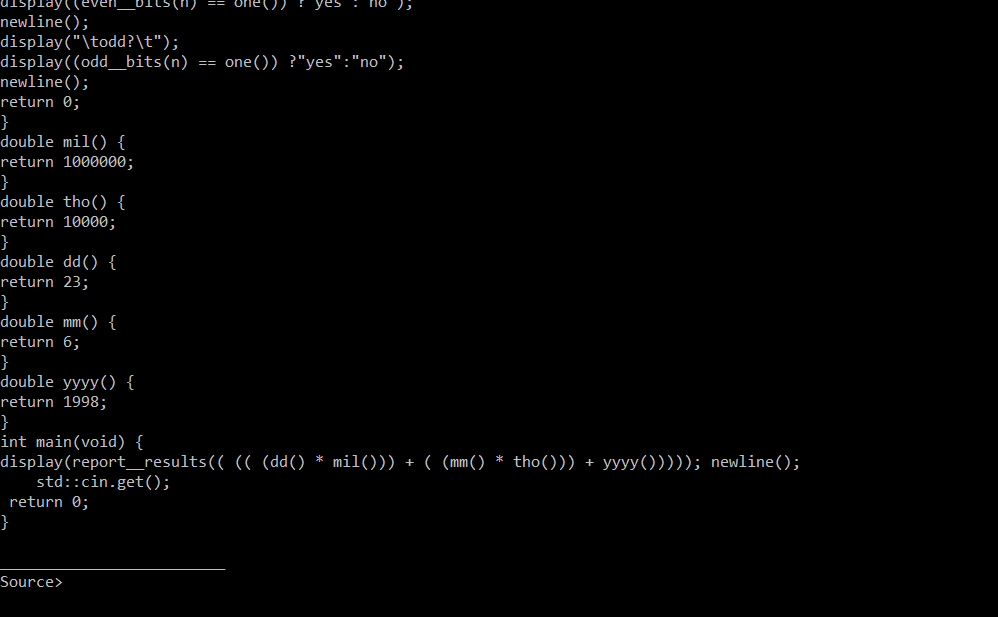
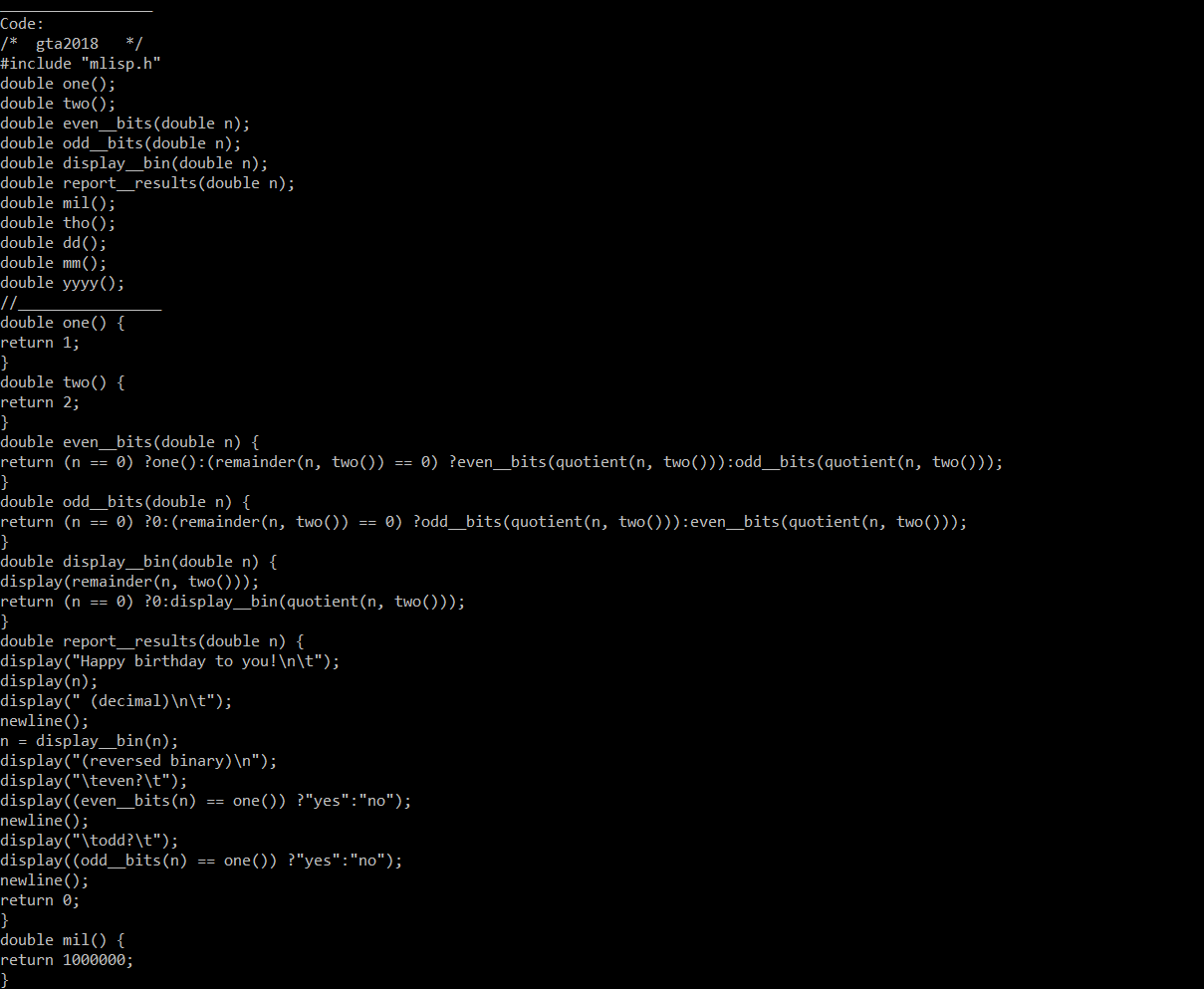
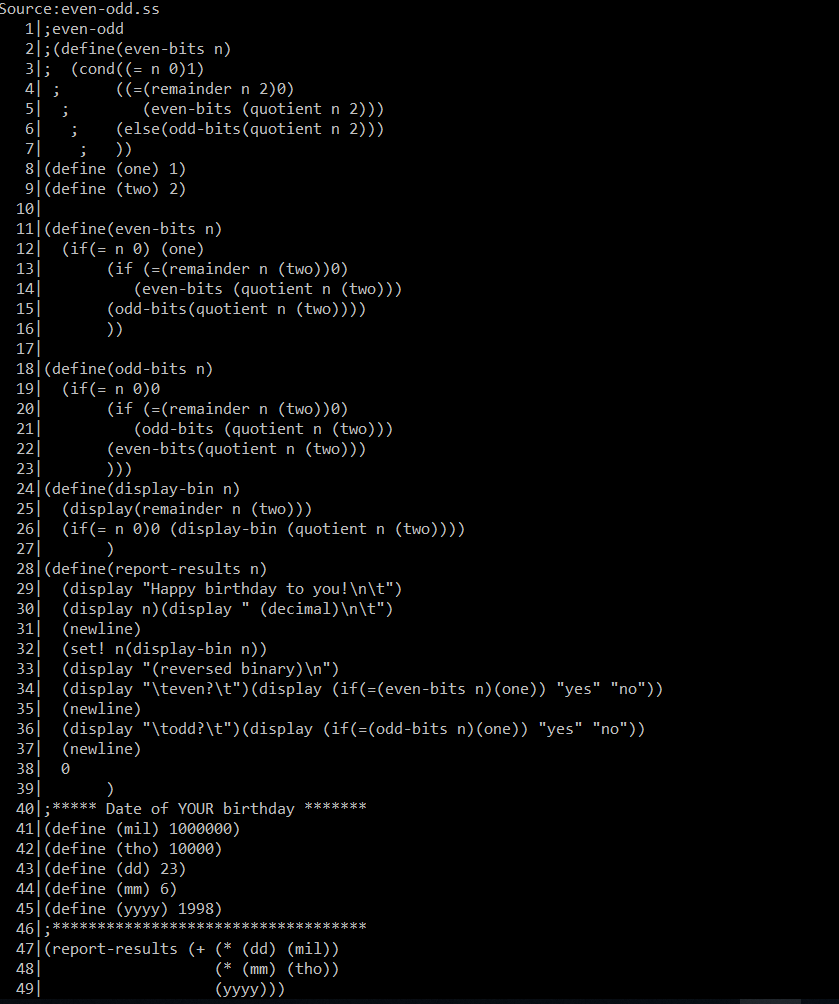
**13. Локальные переменные.**

**\*13.2 Только параметры процедур.**

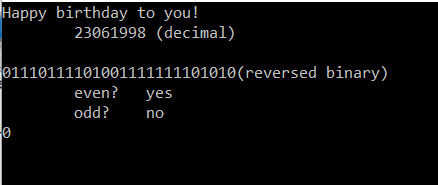
**(define(f a) (set! a pi)a) (f 0)**

**Контрольная задача №1.**

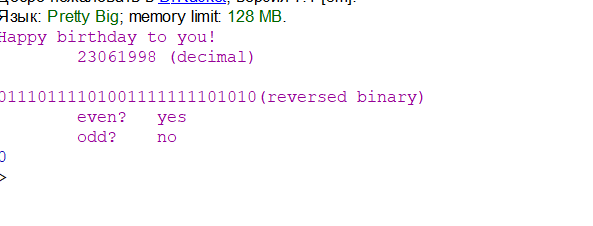
**Полный протокол трансляции без трассировки (крупный белый шрифт на ярком черном фоне).**

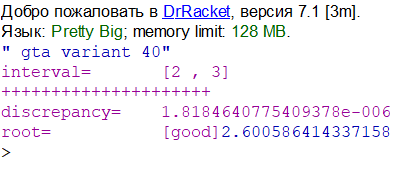
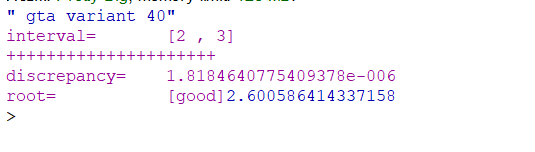
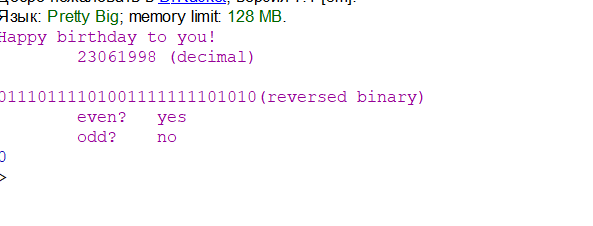
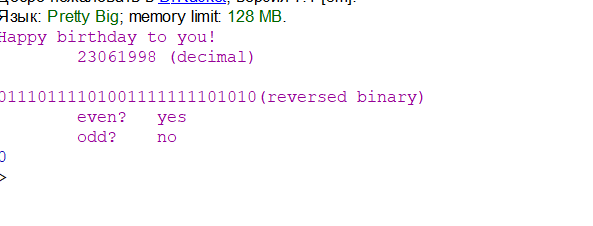


**Протокол запуска задачи на С++.**



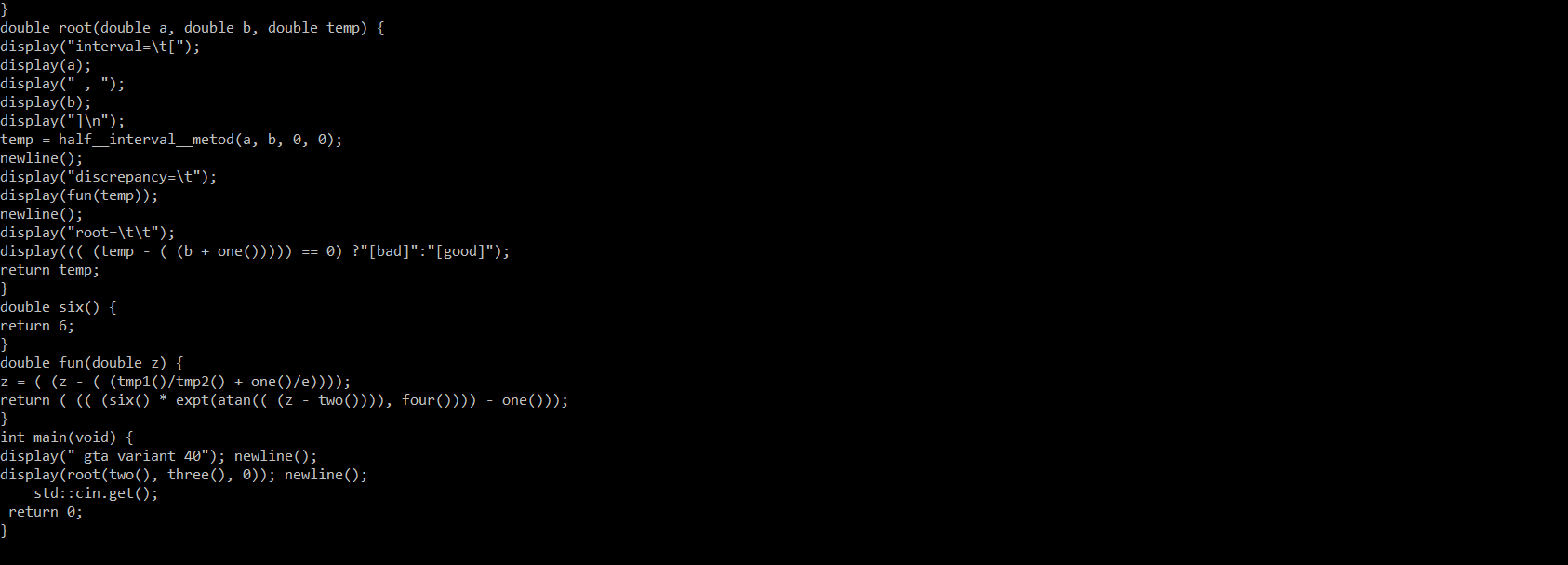
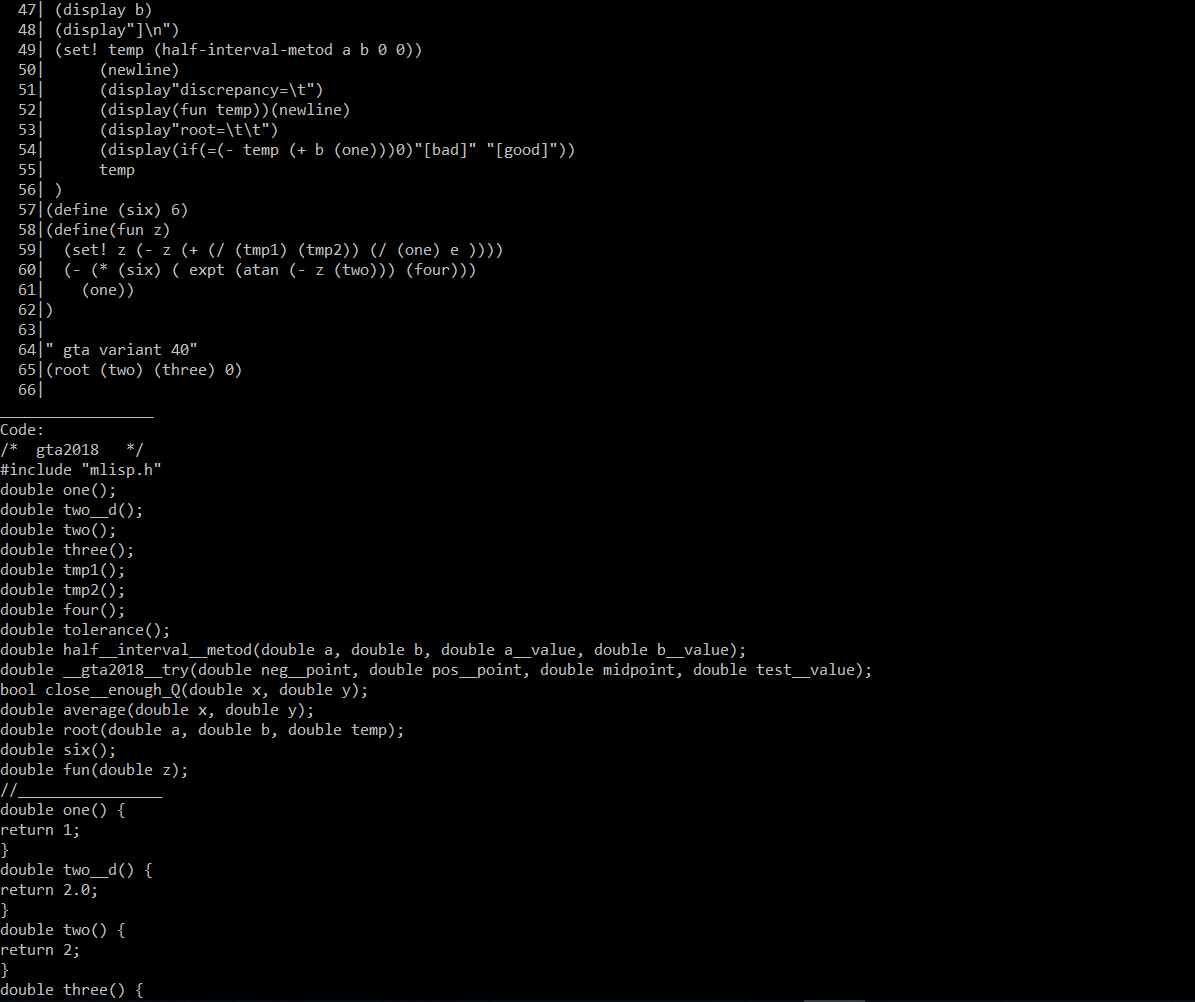
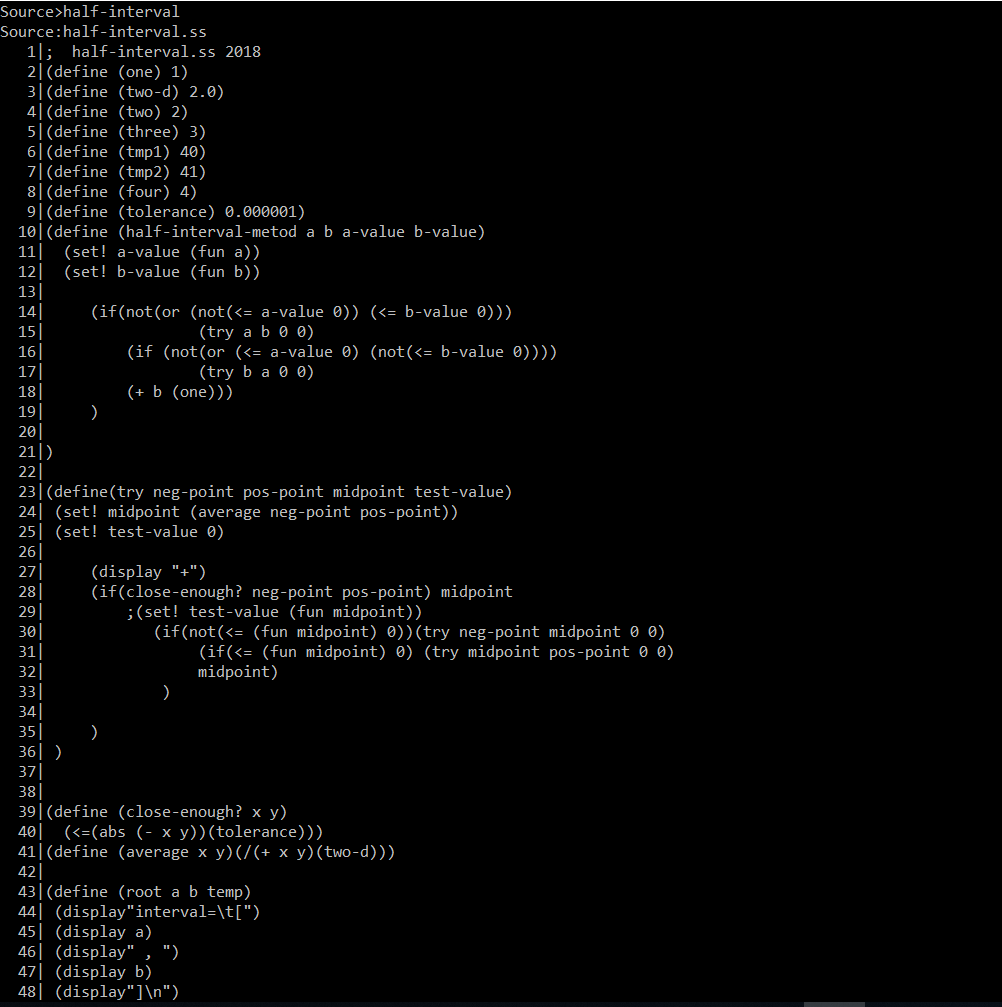
**Протокол запуска задачи на Лиспе.**





**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).**

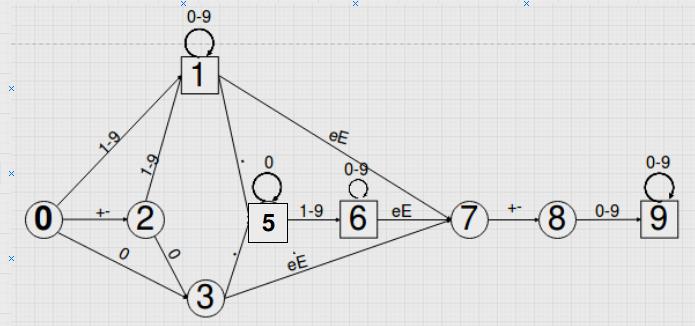
**Протокол запуска задачи на С++.**

**Протокол запуска задачи на Лиспе.**

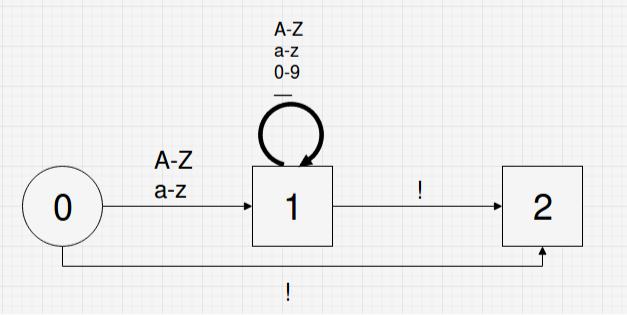
**Распечатка файла code-gen.cpp.**

**Диаграммы автоматов из лабораторной работы №5 для токенов $dec, $id, $idq.**

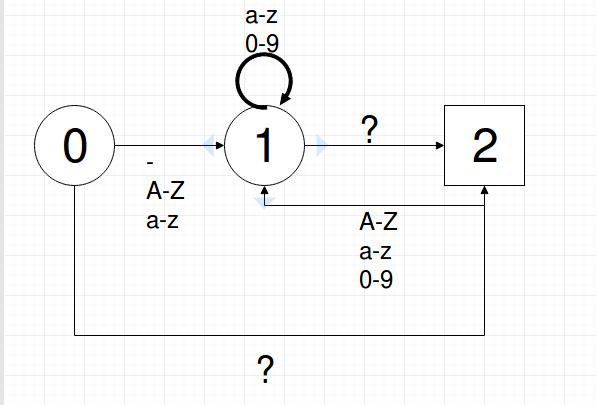
**$dec: 4. Дополнительных ограничений нет.**



**$id: 2. ‘!’ можно использовать только один раз, и только на последнем месте.**



**$idq: 3. знак ‘–‘ не может стоять непосредственно после ‘?’.**



**Вывод:**

* **данной курсовой работе я применил все навыки, которые получил за данный курс.**
* **итоге удалось разработать мини синтаксический транслятор из языка MicroLisp в C++. Перечисление полученных навыков:**

**1. Построение конечных автоматов для распознования токенов языка: $id, $idq, $dec.**

**2. Писать код для простых задач на языке MicroLisp и вручную транслировать его в C++.**

**3. Строить дерево разбора для заданных продукций грамматики согласно варианту задания, а также реализовывать их для транслятора.**

**Согласно варианту e04 грамматика языка MicroLisp сильно ограничена, но даже этого достаточно для написания простых программ.**

**Первым шагом написания курсового проекта стояла задача переписать 1-3 л.р. под свою грамматику из файла GrammaFeatures. Столкнулся с проблемой объявления глобальных и локальных переменных — решение оказалось очевидным: реализовать функции (константы) и передавать переменные через параметры процедуры. В итоге mlispgen проверил все три фала на языке MicroLisp(согласно моей гамматике) на наличие ошибок, которых как ни странно не было. Следующий шаг — дописывание грамматики в файле code-gen.cpp. Было довольно просто — благодаря комментариям, быстро ориентировался и находил зависимости, правда, не без ошибок.**

**Ошибки были банальными:**

**1. Лишние скобки в условии cond.**

**2. Неправильное создание констант через функции.**

**3. Тернарный оператор(знаков «?» было больше чем «:»).**

**4. Сложение не работало после деления. После исправления ошибок программа mlispgen сгенерировала компилируемые без ошибок файлы на языке C++.**

* **результате полученные программы отработали эквивалентно программам написанных на MicroLisp. Поставленная задача для первой части курсового проекта была выполнена в полном объеме.**