;синтаксис транслятора: каждая строка начинается с имени команды. в одной строке одна команда.
;name_command_0 ...//всё дальнейшее задаётся синтаксисом команды
;строки состоящие только из ';', табуляции или пробелов игнорируются
;67 уникальных команд не считая nop'a и src'a!

символ ';' - обозначение конца команды, всё, что после - игнорируется прим. если не заслешен
символ '\n' - строго конец строки, содержащей команду прим. если не заслешен
nop - отсутствие команды
<nop>[;]<endl>
arr - команда, задающая базовое множество
перед каждым сиволом строки, описывающей элемент множества, обязателен бэкслеш
<arr> <name> <string>,<namestr>[,<string>][;]<endl>
<arradd> <string>,<namestr>[,<string>][;]<endl> добавляет к множеству ещё несколько строк
<arrrem> <string>,<namestr>[,<string>][;]<endl> удаляет из заданного множества заданные строки
bor - команда, задающая структуру, которая способна выдавать
уникальные последовательные номера строкам
перед каждым сиволом строки обязателен бэкслеш
<bor> <name> <string>,<namestr>[,<string>][;]<endl>
<boradd> <name> <string>,<namestr>[,<string>][;]<endl> добавляет к множеству ещё несколько строк
<borrem> <name> <string>,<namestr>[,<string>][;]<endl> удаляет из заданного множества заданные строки
<borrem> <name> <num1>,<num2>[,<num3>][;]<endl> удаляет из заданного множества строки с заданными номерами.
номера строки и строки-литералы можно совмещать!
<borgts> <name> <stringvar> <num>[;]<endl> возвращает строку с конкретным номером
<borgtn> <name> <num> <stringvar>[;]<endl> возвращает номер конкретной строки.
если строки не было, то добавит и присвоит ей номер!
номера присваиваются в порядке возрастания, а после удаления элемента номера пересчитываются.
obj - команда создания объекта. объект состоит из частей
которые задаются множествами и/или объектами
каждая часть отделяется от следующей запятой.
итоговый объект это набор частей, для каждой из которой выполнено условие
что данная часть состоит только из одного
из указанных множеств данной части, повторённая maxcnt раз
перечисление множеств идёт в порядке приоритета
<obj> <name> maxcnt <namearr>,maxcnt <namearr1> maxcnt <namearr2>, maxcnt <nameobj>, ..., ..., ...[;]<endl>
env - объявляет новый контекст исполнения с заданным именем. В новый контекст
входят все последующие команды и объявления. Команда без указания имени контекста
обозначает конец последнего начатого контекста. Контекст это названный кусок кода.
все имена в нём, кроме переходов к другим контекстам известны.
env это самостоятельная единица. env не содержит в себе другие env.
<env> <name>[;]<endl>
<endenv>[;]<endl> конец контекста
run - приостанавливает текущий контекст и запускет новый.
<run> <name>[;]<endl>
chg - завершает текущий контект, запуская вместо него новый.

<chg> <name>[;]<endl>
met - устанавливает метку на следующую команду в текущем контексте
<met> <name>[;]<endl>
jmp - осуществляет переход к метке внутри контекста
<jmp> <name>[;]<endl>
end - прекращает исполнение текущего контекста, переходя к последнемку приостановленному
<end>[;]<endl>
psh - кладёт на вершину стека переменную. все переменные это ссылки на реальное значение.
учёт числа ссылок ведётся автоматически. объекты без ссылок самоуничтожаются.
<psh> <name>[;]<endl>
cln - создаёт копию исходного объекта в куче.
<cln> <name> <oname>[;]<endl>
var - новая ссылка на объект.
<var> <rname> <name>[;]<endl>
ссылки типизированы, каждая ссылка знает тип хранимогог значения.
pop - достаёт ссылку на объект в указанную ссылку
с вершины стека. если в стеке ничего нет вернёт пустую ссылку.
<pop> <rname>[;]<endl>
mth - работает с целочисленными переменными. арифметика длинная.
Поддерживает сложение, вычитание, умножение и деления с и без остатка.
Запись операций ведётся в обратной польской нотации. количество знаков до тысячи разрядов.
Внимание! Не вызывать переполнения!!!
*<mth> <name> 1 2 + 3 * 4 5 6 - - / 2 %[;]<endl>*
set - команда, устанавливающая зачение флага в определённое 32 разрядное знаковое значение.
Количество флагов 30 разрядное число без знака.
для скорости работы не используйте более 100К первых флагов.
<set> <number of flag> <flag value>[;]<endl>
get - команда получает значение флага в целочисленную переменную.
<get> <name> <number of flag>[;]<endl>
tst - команда ветвления. способна делать различные проверки.
команда ветвления начинается с <tst> далее один из методов проверки
nil - выполнит следующую команду, если ссылка, поданная ему на вход является нулевой.
<tstnil> <rname>[;]<endl>
<next command>[;]<endl>
lrgoreql - выполнит следующую команду, если число A меньше числа B
<tstlss> <A> <B>[;]<endl>
<next command>[;]<endl>
lrg - выполнит следующую команду, если число A больше числа B
<tstlrg> <A> <B>[;]<endl>
<next command>[;]<endl>
eql - выполнит следующую команду, если число A равно числу B
<tsteql> <A> <B>[;]<endl>
<next command>[;]<endl>
stk - выполнит следующую команду, если стек не пуст
<tststk>[;]<endl>
<next command>[;]<endl>
is где * это*
a для множества

o для объекта
i для числа
f для фрейма
s для строки
c для файла
v для вектора
выполнит следующую команду, если то, о чём мы спррашиваем является тем, что нужно
<tstis> <anything>[;]<endl>*
<next command>[;]<endl>
flg - проверяет, что флаг не ноль.
<tstflg> <number of flag>[;]<endl>
abvoridt - выполнит следующую команду, если строка A меньше строки B(лексикографически)
<tstblw> <A> <B>[;]<endl>
<next command>[;]<endl>
abv - выполнит следующую команду, если строка A больше строки B(лексикографиески)
<tstabv> <A> <B>[;]<endl>
<next command>[;]<endl>
idt - выполнит следующую команду, если строка A индентична строке B
<tstidt> <A> <B>[;]<endl>
str - создаёт заданную строку.
<str> <name> = <\><symbol><\><symbol>...[;]<endl>
mts - создаёт строку из числа
<mts> <namestr> <name math>[;]<endl>
fts - фрейм в строку.
<fts> <name str> <name frame>[;]<endl>
stm - строка в число
<stm> <name math> <mame str>[;]<endl>
vct - создаёт массив указанной длины
<vct> <namevct> <math or numb>[;]<endl>
<vctpshbck> <elem>[;]<endl> положить в конец
<vctpshfrt> <elem>[;]<endl> положить в начало
<vctpopbck> <var>[;]<endl> достать с конца
<vctpopfrt> <var>[;]<endl> достать с начала
<vctset> <num> <elem>[;]<endl> установить элемент
<vctget> <var> <num>[;]<endl> взять значение
<vctgsz> <num>[;]<endl> получить размер
<vctrsz> <num>[;]<endl> изменить размер
txt - преобразования над строками.
<txtlen> <math> <stringvar>[;]<endl> длина строки
<txtctn> <str1> <str2var>[;]<endl> конкатенация строк
<txtfnd> <vectorvar> <stringvar> <substr>[;]<endl> находит все индексы вхождения подстроки в строкеи записывает их в массив
<txtspl> <vectorvar> <stringvar> <math>[;]<endl> разрезает строку на две [0,math-1] и [math, len]
<txtdel> <stringvar> <math1> <math2>[;]<endl> вырезает с math1 элемента math2 штук символов
prt - извлекает указанную часть фрейма
<prt> <var> <frm> <math>[;]<endl> В var считывается часть. frm - исходный фрейм. math - номер извлекаемой части.
opn - открывает файл
<opnrdf> <var> <stringvar>[;]<endl> - открывает файл str для чтения

<opnwrf> <var> <stringvar>[;]<endl> - открывает файл для записи
cls - закрывает файл
<cls> <var file>[;]<endl>
nxt prv - смещение указателя на заданное число сиволов вперёд и назад по файлу
<nxt> <var file> <math>[;]<endl>
<prv> <var file> <math>[;]<endl>
wrt - записывает в файл строку, число, фрейм, множество,
текстовое представление объекта, имя файла, текстовое представление вектора
<wrt> <var file> <any thing>[;]<endl>
scn - считываем из объекта
<scndef> <stringvar> <math> <filevar>[;]<endl> - не более math символов в строку <stringvar> из файла.
<scnfltfil> <frame> <object> <filevar>[;]<endl> - попытка считать из файла объект object.
если случилась неудача, то исполнится следущая команда, иначе следущая команда будет пропущена.
<scnfltstr> <frame> <object> <stringvar>[;]<endl> - попытка считать из строки объект object.
если случилась неудача, то исполнится следущая команда, иначе следущая команда будет пропущена.