Реализуйте приложение-интерпретатор с настраиваемым синтаксисом и возможностями отладки программ, написанных для реализованного интерпретатора.
Для настройки интерпретатора приложению через аргументы командной строки подается файл с описанием имён инструкций и их синтаксиса. Файл настроек содержит сопоставления операций, которые может выполнить интерпретатор, и их псевдонимов, которые будут использованы в программах, которые будут поданы на вход. Файл настроек может содержать однострочные комментарии, которые начинаются с символа #.
Интерпретатор оперирует 32-х разрядными целочисленными беззнаковыми переменными, имена которых могут содержать один и более символов (в качестве символов, входящих в имена переменных, допускаются символы букв латинского алфавита, арабских цифр и подчерка (‘_’); имя переменной не может начинаться с символа цифры; длина имени переменной произвольна; прописные и строчные буквы не отождествляются).
Команды, которые могут быть выполнены интерпретатором:

- Унарные:
  - not - поразрядная инверсия;
  - input -ввод значения из стандартного потока ввода в системе счисления с основанием base_input;
  - output -вывод значения переменной в стандартный поток вывода в системе счисления с основанием base_output;
- Бинарные:
  - add - сложение;
  - mult - умножение;
  - sub - вычитание;
  - pow - возведение в целую неотрицательную степень по модулю 2^{32} алгоритмом быстрого возведения в степень по модулю;
  - div - целочисленное деление;
  - rem - взятие остатка от деления;
  - xor - поразрядное сложение по модулю 2;
  - and - поразрядная конъюнкция;
  - or - поразрядная дизъюнкция;
  - = - присваивание значения переменной или её инициализация значением выражения или константой в системе счисления с основанием base_assign.

В рамках инструкции, обрабатываемой интерпретатором, допускается выполнение нескольких команд. Пример файла (при base_assign = 16):

```
var_1 = 1F4;
var_2 = mult(var_2, 4);
Var3 = add(div(var_2, 5) , rem(var_1, 2));
print(Var3);
```

Для каждой из вышеописанных команд можно задать синоним в файле настроек интерпретатора. Для этого на отдельной строке файла необходимо сначала указать

оригинальное название команды, далее через пробел - синоним. Если одна и та же команда в файле настроек заменяется синонимом несколько раз, в результате должен быть применён только последний встреченный синоним; при задании нескольких синонимов для одной и той же команды в одном файле настроек оригинальное написание команды не сохраняется на уровне файла настроек. Если для команды не задаётся синоним, она сохраняет своё оригинальное написание.

Помимо синонимов для команд, выполняемых интерпретатором, необходимо реализовать возможность конструирования синтаксиса инструкций относительно файла настроек:

- Сохранение результатов выполнения операций:
  - left= - при наличии этой инструкции в файле настроек, в обрабатываемом файле переменные, в которые будет сохранён результат выполнения операции, должны находиться слева от операции. Пример:
    Var=add(Smth,OtheR);
  - right= - при наличии этой инструкции в файле настроек, в обрабатываемом файле переменные, в которуе будет сохранён результат выполнения операции, должны находиться справа от операции. Пример:
    add(Smth,OtheR)=Var;
- Взаимное расположение операндов и операции, выполняемой над операндами:
  - Для унарных операций:
    - op() - при наличии этой инструкции в файле настроек, аргумент операции находится после операции и обрамляется скобками. Пример:
      result=operation(argument);
    - ()op - при наличии этой инструкции в файле настроек, аргумент операции находится перед операцией и обрамляется скобками. Пример:
      result=(argument)operation;
  - Для бинарных операций:
    - op() - при наличии этой инструкции в файле настроек, аргументы операции находятся после операции и обрамляются скобками. Пример:
      result=operation(argument1,argument2);
    - (op) - при наличии этой инструкции в файле настроек, первый аргумент операции находится перед операцией, второй аргумент операции находится после операции. Пример:
      result=argument1 operation argument2;
    - ()op - при наличии этой инструкции в файле настроек, аргументы операции находятся после операции и обрамляется скобками. Пример:
      result=(argument1,argument2)operation;

Пример файла настроек:

right= #это комментарий

(op)
add sum
#mult prod и это тоже комментарий
[sub minus
pow ^ и это...]
div /
rem %
xor <>
xor ><
#xor <>
input in
output print
= ->

Значения base_assign, base_input, base_output задаются при помощи передачи значений в аргументы командной строки, могут находиться в диапазоне [2..36] и имеют значение по умолчанию равное 10 (при отсутствии в командной строке).
Разделителем между инструкциями в обрабатываемом интерпретатором файле является символ “;”. Сепарирующие символы (пробелы, табуляции, переносы строк) между лексемами могут присутствовать произвольно, различий между прописными и строчными буквами нет. Также в тексте программ могут присутствовать однострочные комментарии, начинающиеся с символа # и заканчивающиеся символом конца строки или символом конца файла; многострочные комментарии, обрамляющиеся символами ‘[‘ и ‘]’, вложенность многострочных комментариев произвольна.
Входной файл для интерпретатора подаётся через аргументы командной строки. Реализуйте обработку интерпретатором инструкций из входного файла. При завершении работы интерпретатор должен “запомнить” последний файл настроек, с которым работал, и при следующем запуске работать с теми же настройками, если это возможно.
В качестве аргумента командной строки приложению можно подать флаг “--debug”, “-d”, “/debug” (перечисленные флаги эквивалентны). При наличии одного из таких флагов в аргументах командной строки, однострочный комментарий с текстом “BREAKPOINT” (без кавычек) интерпретируется приложением как точка останова: выполнение интерпретатором кода из входного файла на момент встречи этого комментария приостанавливается, и пользователь начинает взаимодействовать с программой в интерактивном режиме посредством диалога. Интерактивный диалог должен предложить пользователю выбор из нескольких возможных действий:

- вывести в стандартный поток вывода значение переменной с именем, считываемым из стандартного потока ввода, в системе счисления с основанием 16, а также снять дамп памяти, в которой хранится это значение (байты значения, записанные в системе счисления с основанием 2, в порядке нахождения в памяти “слева направо”; строковые представления байтов должны сепарироваться одним символом пробела); после выполнения действия интерактивный диалог с пользователем продолжается;

- вывести в стандартный поток вывода имена и значения всех переменных, “известных” (с которыми велась работа) на данный момент исполнения кода; после выполнения действия интерактивный диалог с пользователем продолжается;
- изменить значение переменной, имя которой считывается из стандартного потока ввода, на новое значение, считываемое из стандартного потока ввода как число в системе счисления с основанием 16 (переменная должна быть “известна” на момент выполнения операции); после выполнения действия интерактивный диалог с пользователем продолжается;
- “объявить” переменную, имя которой считывается из стандартного потока ввода и проинициализировать её значением, которое вводится, в зависимости от выбора пользователя в интерактивном диалоге, запускаемом из основного интерактивного диалога, как число в цекендорфовом представлении либо как число записанное римскими цифрами (переменная не должна быть “известна” на момент выполнения операции); после выполнения действия интерактивный диалог с пользователем продолжается;
- “отменить” объявление переменной, имя которой считывается из стандартного потока ввода; дальнейшее использование переменной в коде до её повторного “объявления” (через выполнение инструкции либо через отладчик) должно привести к ошибке интерпретатора (переменная должна быть “известна” на момент выполнения операции); после выполнения действия интерактивный диалог с пользователем продолжается;
- завершить интерактивный диалог с пользователем и продолжить выполнение кода;
- завершить работу интерпретатора.

При возникновении ошибок в рамках интерактивного диалога, программа должна вывести в стандартный поток вывода информацию об ошибке отладчика; состояние переменных при этом не изменяется, интерактивный диалог не завершается, выполнение приложения не завершается.

Взаимодействие с переменными уровня интерпретатора (объявление, присваивание, “отмена” объявления) обеспечьте на основе структуры данных вида “бор”.

Предоставьте текстовый файл с выражениями для реализованного интерпретатора и продемонстрируйте его работу с разными файлами настроек, с разными входными текстовыми файлами с наборами инструкций, а также работу в режиме отладки. Обработайте ошибки времени выполнения инструкций.