Домашнее задание №10

Краткое описание реализации

Интерпретатор реализован с помощью GNU Bison и Flex.

Интерпретатор состоит из лексического анализатора, задаваемого файлом lexer.l, и синтаксического, задаваемого файлом с грамматикой grammar.y. Интерпретатор работает с тиами int и float, названия переменных начинаются с латинской буквы и далее могут содержать заглавные и прописные буквы латинского алфавита, цифры или знаки нижнего подчеркивания. Каждая строчка должна оканчиваться знаком новой строки. Результат логического сравнения - число, равное 1 для результата ИСТИНА, и 0 для _ ЛОЖЬ. Результат любой арифметической операции с целыми числами/перменными - целое число. Если же один из операндом – число или переменная с плавающей точкой, то результат операции - число с плавающей точкой. Интерпретатор печатает опреации присвоения значений переменным. Допускается перезаписывать переменную. Тип переменной определеятся динамически.

Токены хранятся в виде структур token_struct:

```
struct token_struct
{
    bool is_integer;
    std::string str;
    int int_value;
    float float_value;
};
```

token_struct реализован в файлах token_struct.hpp и token_struct.cpp.

Парсер запускаетсятся командой

make run

Тесты запускаются командой

make test

Примеры выходных файлов – в папке ./test_results

Описание грамматики:

Токены
EQ = "=="
LE = "<="
GE = ">="
NE = "!="
NUM - целове число
FLOAT - число с плавающей точкой
VAR - переменная, может быть как целочисленной, так и с плавающей точкой

//программа = последовательность операций
PROGRAM: OPS
;

OPS: OP
| OPS OP

```
;

//каждая операция - выражение на одной строке или пустая строка
OP:     '\n'
|       EXPR '\n'
;

//присваивание переменной значения
EXPR:   EXPR1
|       VAR '=' EXPR

//операции сравнения
EXPR1:  EXPR2
|       EXPR1 EQ EXPR2
|       EXPR1 LE EXPR2
|       EXPR1 GE EXPR2
|       EXPR1 NE EXPR2
|       EXPR1 '>' EXPR2
|       EXPR1 '<' EXPR2
;

//арифметические операции
EXPR2: TERM
|       EXPR2 '+' TERM
|       EXPR2 '-' TERM
;

TERM:   VAL
|       TERM '*' VAL
|       TERM '/' VAL
;

//скобки, отрицание и унарный минус
VAL:    NUM
|       FLOAT
|       '-' VAL
|       '!' VAL
|       '(' EXPR ')'
|       VAR
;
```