Задание 2. Интерпретатор Forth-like языка
(30 ноября 10:00)

Написать интерпретатор языка, который поддерживает часть функциональности языка Forth

Описание языка. Базовая версия (15 баллов)

Числа

Поддерживаются только целые числа, границы числа совпадают с границами типа int в C++ на вашей платформе

При подаче числа в интерпретатор число кладется на стек:

> 3
< ok

Арифметические операции

Поддерживаемые операции: *** / + - mod**

Все эти операции берут в качестве правой операнды вершину стека, в качестве левой - второе число на стеке, вычисляют результат и кладут на стек. Если на стеке недостаточно чисел или если операция нелегальна, то должно выбрасываться соответствующее исключение и печататься ошибка (числа снимаются со стека)

Примеры

1) > 3 2 +
   < ok
2) > 3
   < ok
   > 3
   < ok
   > *
   < ok
3) > 3
   < ok
   > 0 /
   < “Error: division by zero”

Встроенные команды

dup -скопировать вершину стека и положить на стек
drop - снять верхнее число со стека
. - напечатать верхнее число на стеке, снять со стека
swap - поменять местами верхнее и второе числа на стеке
rot - циклически сдвинуть три верхние числа.
например, стек до: 4 1 2 3, стек после: 4 3 1 2
over - скопировать второе число и положить копию над верхним.
например, стек до: 3 2 1, стек после: 3 2 1 2
emit - распечатать верхнее число на стеке как ascii код и снять со стека.
например, 65 emit напечатает A
cr - перенос строки. например, *100 200 . cr .* напечатает 200\n100

Логические операции

Поддерживаемые операции: **> < =**

Все эти операции берут в качестве правой операнды вершину стека, в качестве левой - второе число на стеке, вычисляют результат и кладут на стек. Если на стеке недостаточно чисел или если операция нелегальна, то должно выбрасываться соответствующее исключение и печататься ошибка (числа снимаются со стека).
Результат операции: 0, если результат операции false; 1, если true

Пример
1) > 3 2
   < ok
   > =
   < ok
   > .
   < 0

Печать строк

Команда вида **.”** STRING_CONTENT**”** печатает заданную строку (без кавычек).

Пример
1) > .” Foo”
   < Foo

Описание языка. Продвинутая версия (+ 5 баллов)

Условный оператор

Поддерживается два формата:

1) **if** THEN_BRANCH **then ;**
   Если на стеке лежит 0, то ничего не происходит, если любое другое число, то выполняется THEN_BRANCH.Точка с запятой после команды обязательна
2) **if** THEN_BRANCH **else** ELSE_BRANCH **then ;**
   Если на стеке лежит 0, то выполняется ELSE_BRANCH, если любое другое число, то выполняется THEN_BRANCH. Точка с запятой после команды обязательна

Цикл

do LOOP_BODY **loop ;**

Внутри цикла можно использовать специальное ключевое слово **i** - текущий индекс в цикле. Вершина стека до исполнения цикла - начальное значение **i**, второе число - (конец цикла + 1). оба числа снимаются со стека.

Пример
> 10 0 i .
< 0 1 2 3 4 5 6 7 8 9

Команды

Пользователь может объявлять свои команды:
: **command_name** COMMAND_BODY
После этого можно пользоваться командой просто напечатав **command_name**.
Команды можно переопределять

Пример:
> : foo 0 **do** .” Foo” **loop** ;
< ok
> 3 foo
< Foo Foo Foo
> 5 foo
< Foo Foo Foo Foo Foo

Переменные

Переменная - это именованная область данных. Возможные операции:

1) **variable** VAR_NAME - объявление переменной, по умолчанию инициализируется 0

2) VAR_NAME - положить адрес переменной на стек
3) **@** - разыменовать переменную по адресу на стеке и положить значение на стек
4) **!** - записать второе число на стеке в переменную по адресу равному верхнему числу на стеке, снять оба числа со стека

Массивы

Массив - это область данных, вмещающая в себя несколько переменных. Определены следующие операции:

1) **variable** ARRAY_NAME - объявление массива, длина 1
2) N cells allot - выделить еще (N * cells) места. следующая объявленная переменная будет с указанным сдвигом

Требования

- Код должен быть покрыт тестами https://github.com/google/googletest
- Публичные методы должны быть документированы (на английском)
- Программа должна собираться с помощью cmake
- В программе не должно быть утечек памяти
- Экземпляры команд должны быть обернуты в умные указатели
- В программе должны быть определены пользовательские исключения
- Рекомендуется использовать стандартные коллекции (std::map, std::stack...)

Опциональные задания

- Порешать задачи из Starting Forth используя собственный интерпретатор
- Добавить поддержку чтения скрипта из файла (также добавить комментарии в язык)

Полезные ссылки

- https://skilldrick.github.io/easyforth/
- https://www.forth.com/starting-forth/
- https://craftinginterpreters.com/contents.html