федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский УНИВЕРСИТЕТ информационных технологий, механики и оптики

**Факультет программной инженерии и компьютерной техники**

**Кафедра «Вычислительной техники»**

**КУРСОВАЯ РАБОТА «Разработка интерпретатора языка Forth»**

**ПО ДИСЦИПЛИНЕ «Системное программное обеспечение»**

Выполнила: Орлова Кристина Александровна

Группа: P3202

Преподаватель: Дергачев Андрей Михайлович

Санкт-Петербург

2017/2018

**Часть 1**

Задание

1. Изучите файл README.md в репозитории Forthress.
2. Изучите программы из предыдущей секции.
3. Напишите программу, проверяющую число на четность. Вы можете подавать программу на вход скрипту start и она будет выполняться при запуске. Слив его содержимое с потоком ввода с помощью cat вы можете получить эффект аналогичный предзагрузке этого файла.
4. Напишите программу, проверяющую число на простоту.
5. Напишите программу так, чтобы она выделяла с помощью allot ячейку в памяти, записывала туда результат и возвращала её адрес. Обратите внимание, вы не обязаны делать все эти действия в одном и том же слове. Делайте максимально короткие слова и комбинируйте их.
6. Создать строку в куче в Forthress можно с помощью следующего синтаксиса: m" string". Слово prints печатает строку по указателю. Память в куче должна освобождаться с помощью heap−free. С помощью heap−show можно получить диагностическую информацию о куче. С помощью ? можно получить диагностическую информацию о любом числе или адресе. Например, попробуйте ввести:   
   42 ? ' dup ? m" hello, world" ?
7. Прочитайте про слова c@, c!. Напишите слово, которое принимает указатели на две строки и возвращает их конкатенацию. Память под строку-результат вы можете выделить с помощью слова heap−alloc, которое принимает количество байт для выделения и Протестируйте полученное слово. Нет ли утечки памяти? Вторая часть задания организована по вариантам. Ваш вариант подсчитывается с помощью следующей функции, применённой к вашей фамилии:  
   ( str − num )   
   : string−hash   
   0 >r ( init accumulator )   
   repeat   
   dup c@ ( stacks: str char, acc )   
   dup if ( not end of the line )   
   r> 13 \* + 65537 % ( iteration of hash computations )   
   >r 1 + 0   
   else ( end of line )   
   drop drop r> 1   
   then   
   until  
    ;

Это слово уже определено в файле hash.frt. Номер варианта подсчитывается так: от вашей фамилии берётся хэш, затем вы берёте значение хэша по модулю 3.   
Варианты:

0 Напишите слово, которое для положительного числа построит последовательность Коллатца. Последовательность Коллатца строится так:

• Если текущий член последовательности делится на два, то делим его пополам.

• Иначе умножаем его на 3 и прибавляем 1.

Последовательность Коллатца всегда приходит к единице, но никто не знает, почему.

1 Напишите слово, которое проведёт проверку числа на примарность. Примарным называется число, которое представляется в виде произведения простых без повторов. Например, число 4 не примарное, т.к. 4 = ; 20 – не примарное число, т.к. 20 = · ; 15 – примарное число, т.к. 15 = · .

2 Напишите слово, которое найдёт произведение всех простых делителей числа (его радикал). Например, для числа 20 это 10, т.к. 20 = · , его простые делители 2 и 5.

Выполнение

• Проверка на четность (задание №3)

: const 7 ;

: put\_rest % ;

: check if ." Not even number" else ." Even number" then ;

const 2 put\_rest check

• Проверка на простоту (задание №4)

: const 19 ;

: get\_positive >r r@ 0 < if r> -1 \* else r> then ;

: loop 2 >r repeat const r@ % if r> 1 + >r r@ else const then const = until r> ;

: write\_answer if ." This is a composite number" else ." This is a prime number" then cr ;

: get\_result >r const r> / get\_positive 1 - write\_answer ;

: check\_zero\_one const 0 = if -1 else const get\_positive 1 = if -1 else 1 then then ;

: start check\_zero\_one 1 = if loop get\_result else 0 write\_answer then ;

start

• Выделение памяти (задание №5)

: const 19 ;

: get\_positive >r r@ 0 < if r> -1 \* else r> then ;

: loop 2 >r repeat const r@ % if r> 1 + >r r@ else const then const = until r> ;

: write\_answer if 0 8 allot >r r@ ! r> else 1 8 allot >r r@ ! r> then .S ; ( 1 if it's prime, 0 if not )

: get\_result >r const r> / get\_positive 1 - write\_answer ;

: check\_zero\_one const 0 = if -1 else const get\_positive 1 = if -1 else 1 then then ;

: start check\_zero\_one 1 = if loop get\_result else 0 write\_answer then ;

start

• Конкатенация строк (задание №7, часть 1)

: word1 m" Hello, " ;

: word2 m" world !!!!!" ;

: size1 word1 count ;

: size2 word2 count ;

: size size1 size2 1 + + ;

: allocate size heap-alloc ;

: loop1 word1 size1 0 do >r r@ c@ swap >r r@ c! r> 1 + r> 1 + loop ;

: loop2 word2 size2 0 do >r r@ c@ swap >r r@ c! r> 1 + r> 1 + loop ;

: write\_answer size 1 - - prints .S cr ;

: start allocate loop1 drop loop2 drop write\_answer ;

start

• Построение последовательности Коллатца (задание №7, часть 2, вариант 0)

( str - num )

: string-hash

0 >r ( init accumulator )

repeat

dup c@ ( stacks: str char, acc )

dup if ( not end of the line )

r> 13 \* + 65537 % ( iteration of hash computations )

>r 1 + 0

else ( end of line )

drop drop r> 1

then

until

;

: init m" Орлова" ;

: var init string-hash dup 3 % ;

: current 5 ;

: write\_symbol . ." " ;

: loop repeat dup dup write\_symbol 2 % if 3 \* 1 + else 2 / then dup 1 = until ;

: start var drop drop current loop ;

start . cr

Выводы

В результате выполнения данной работы я познакомилась с новым для себя языком Forthress и научилась на нем писать некоторые элементарные программы: проверку числа на четность, проверку на простоту, выделение памяти с помощью allot, конкатенацию строк, создание последовательности Коллатца для некоторого числа. Так я поняла, что любой человек может написать для себя удобный ему язык на основе ассемблера, который и будет являться разновидностью Forth.