

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Департамент программной инженерии

**КОНСОЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ ОПРЕДЕЛЕНИЯ ЧИСЛА
ЧИСЕЛ ПЕРРЕНА ОТ 1 ДО БЕЗЗНАКОВОГО МАШИННОГО СЛОВА
Пояснительная записка**

Исполнитель:
студент группы БПИ199
_____ Т. С. Кочарян
« _____ » _____ 2020 г.

Москва 2020

Оглавление:

Текст задания.	3
Применяемые расчетные методы.	4
Список используемых источников.	5
Описание работы программы.	6
Тестирование программы.	7
Приложение №1.	11

1. Текст задания.

Разработать программу, определяющую число чисел Перрена от 1 до беззнакового машинного слова.

2. Применяемые расчетные методы.

Для подсчетов n -ого члена последовательности Перрена используется формула $P(n) = P(n - 2) + P(n - 3)$ для $n > 2$ (см. [1]).

Также для работы программы используется массив из 4 элементов, который хранит 4 последовательных члена последовательности чисел Перрена.

Расчеты по проверке на валидность реализованы с помощью команд сравнения и условного перехода.

3. Список используемых источников.

[1] Инструкция по составлению пояснительной записки [Электронный ресурс]. //URL: <http://softcraft.ru/edu/comparch/tasks/mp01/> (Дата обращения: 28.10.2020, режим доступа: свободный)

[2] Статья “Число Перрена”, Wikipedia.org [Электронный ресурс]. //URL: http://bit.do/wikipedia_perrens (Дата обращения: 28.10.2020, режим доступа: свободный)

[3] Perrin sequence (or Ondrej Such sequence) [Электронный ресурс]. //URL: <https://oeis.org/A001608> (Дата обращения: 28.10.2020, режим доступа: свободный)

[4] Программирование на языке ассемблера [Электронный ресурс]. //URL: http://bit.do/assembler_prog (Дата обращения: 28.10.2020, режим доступа: свободный)

4. Описание работы программы.

Программа принимает на вход беззнаковое машинное слово.

Если пользователь вводит 0 или не числовое значение, то выводится сообщение об ошибке.

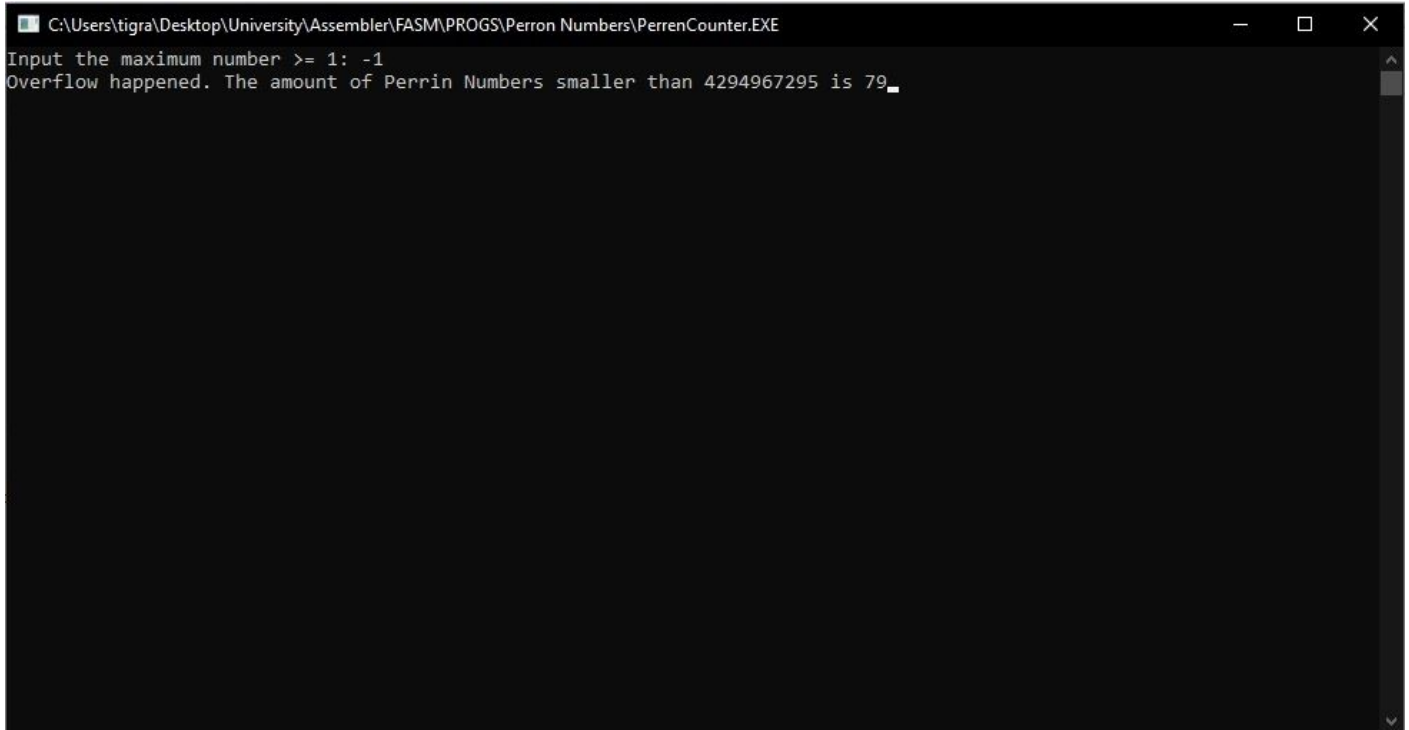
В случае, если это число меньше 0 или больше 2^{32} , в программе предусмотрено взятие этого числа по модулю 2^{32} и продолжение работы программы.

1. Изначальный массив значений 4 последовательных чисел последовательности состоит из 3, 2, 5, 5. Такая реализация обусловлена тем, что некоторые первые члены последовательности повторяются по 2 раза (например, 2, 3 и 5). Если введенное пользователем число < 7 , то на этапе заполнения массива это будет проверено и выведено соответствующее сообщение о количестве чисел Перрена меньше введенного.
2. С помощью регистров `eax`, `ebx`, `ecx` программа сохраняет последние 3 элемента массива (всего в массиве 4 элемента), переносит их значения на первые 3 элемента массива и рассчитывает по формуле $P(n - 2) + P(n - 3)$ значение четвертого элемента массива. Затем, если введенное пользователем число меньше полученного $P(n)$, то программа завершается с сообщением о текущем значении счетчика (в данном случае, это $n-1$). Если же введенное число \geq четвертого элемента массива, счетчик увеличивается на 1 и происходит повтор действий из пункта 2.

5. Тестирование программы.

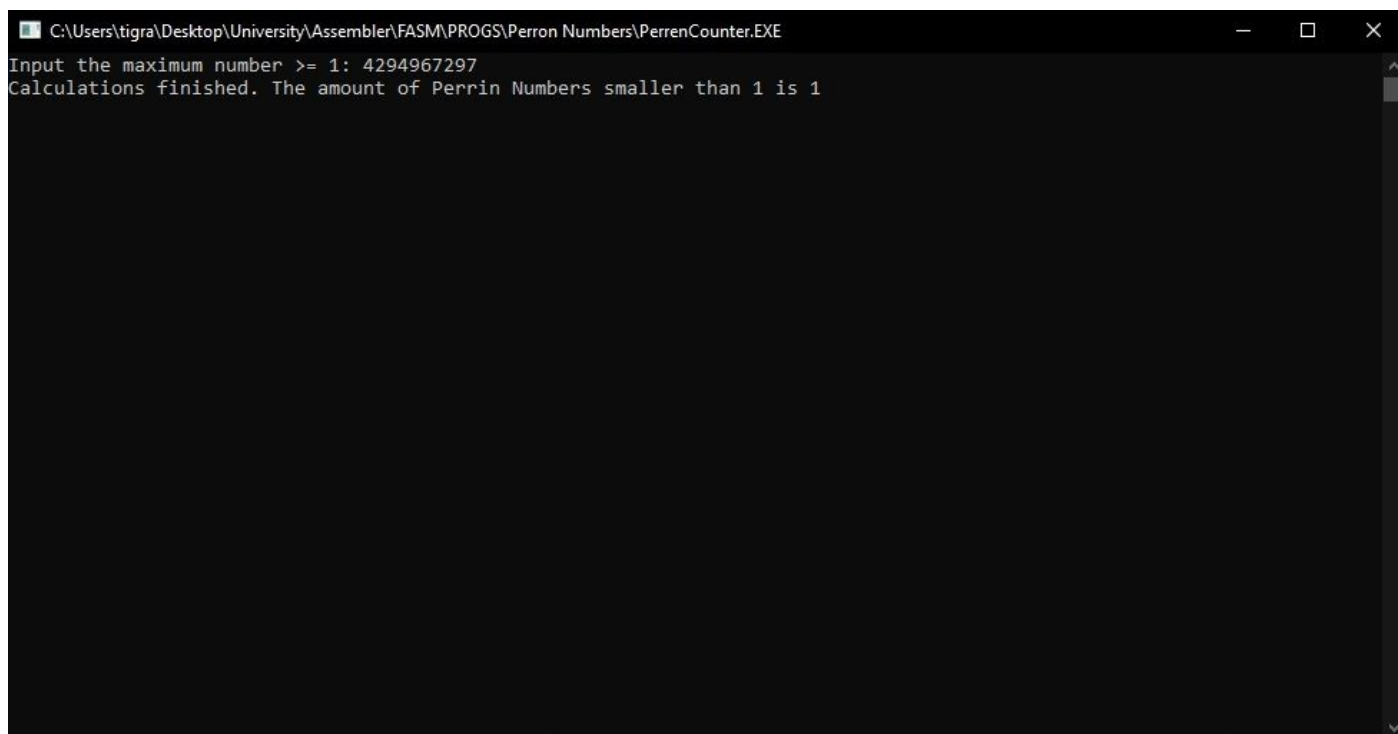
Крайними случаями в данной программе могут быть ввод в консоль числа большего, чем беззнаковое двойное слово или ввод числа < 0 . Для таких случаев в программе предусмотрено взятие этого числа по модулю 2^{32} , чтобы привести к подходящему виду:

Пример 1: Ввод числа < 0 . Если пользователь введёт -1, то по модулю 2^{32} число будет приведено к 4294967295. Таким образом, ответ будет корректным и программа не сломается.



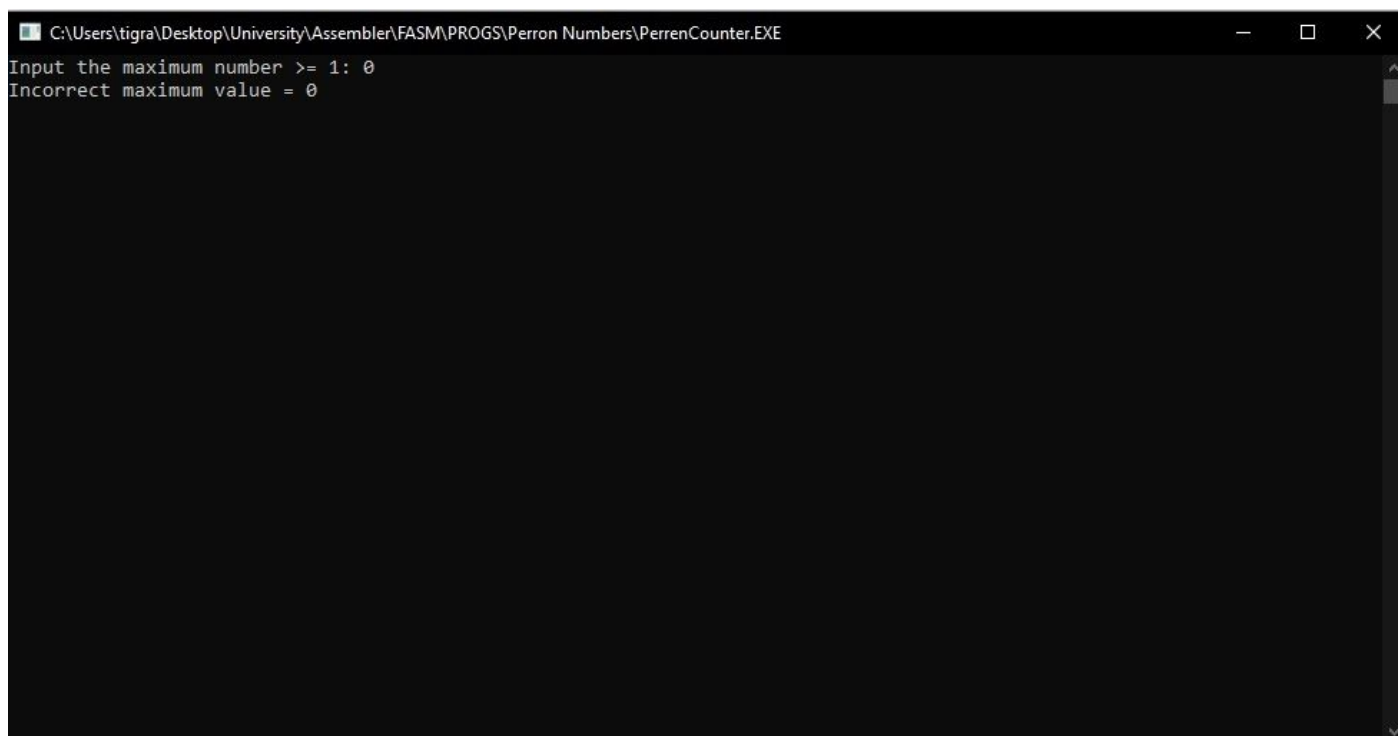
```
C:\Users\tigra\Desktop\University\Assembler\FASM\PROGS\Perron Numbers\PerrenCounter.EXE
Input the maximum number >= 1: -1
Overflow happened. The amount of Perrin Numbers smaller than 4294967295 is 79_
```

Пример 2: В случае ввода пользователем числа большего, чем беззнаковое двойное слово, оно будет взято по модулю 2^{32} . Как видно на примере ниже, $4294967297 \% 2^{32} = 1$. Следовательно, ответ будет 1.



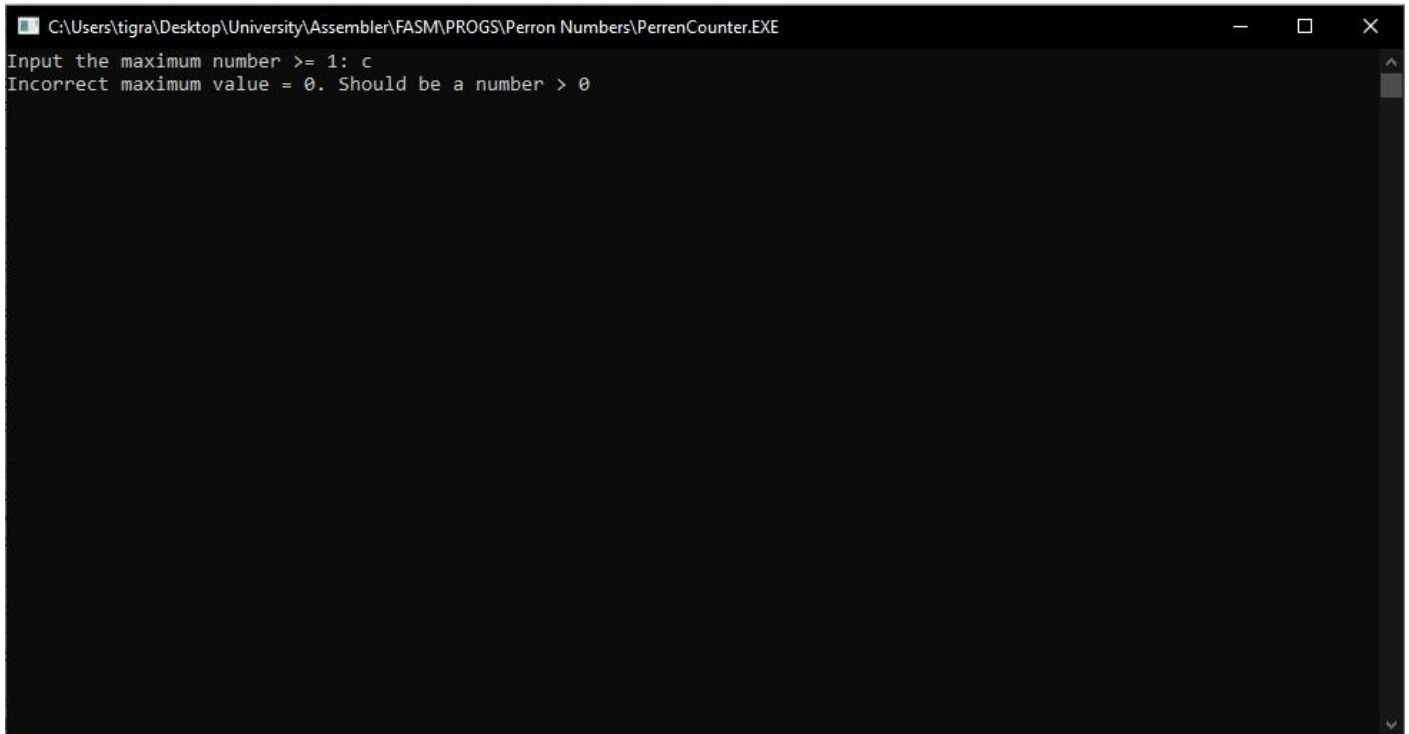
```
C:\Users\tigra\Desktop\University\Assembler\FASM\PROGS\Perron Numbers\PerrenCounter.EXE
Input the maximum number >= 1: 4294967297
Calculations finished. The amount of Perrin Numbers smaller than 1 is 1
```

Пример 3. В случае ввода пользователем 0, программа выведет сообщение об ошибке и завершит свою работу.



```
C:\Users\tigra\Desktop\University\Assembler\FASM\PROGS\Perron Numbers\PerrenCounter.EXE
Input the maximum number >= 1: 0
Incorrect maximum value = 0
```

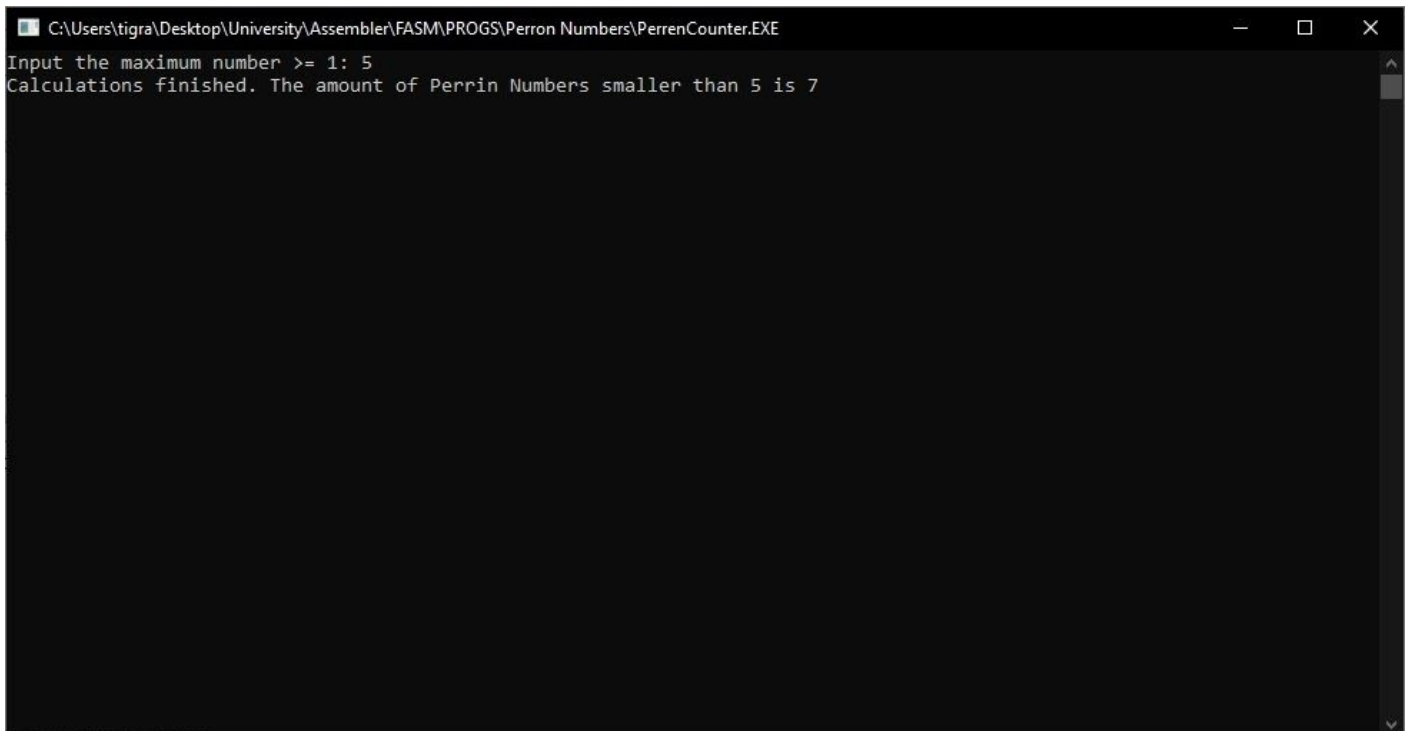

Пример 4. В случае ввода пользователем набора символов, программа выведет сообщение об ошибке.



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\tigra\Desktop\University\Assembler\FASM\PROGS\Perron Numbers\PerrenCounter.EXE. The window has standard minimize, maximize, and close buttons. The command prompt shows the following text: "Input the maximum number >= 1: c" followed by "Incorrect maximum value = 0. Should be a number > 0". The background is black, and the text is white.

```
C:\Users\tigra\Desktop\University\Assembler\FASM\PROGS\Perron Numbers\PerrenCounter.EXE
Input the maximum number >= 1: c
Incorrect maximum value = 0. Should be a number > 0
```

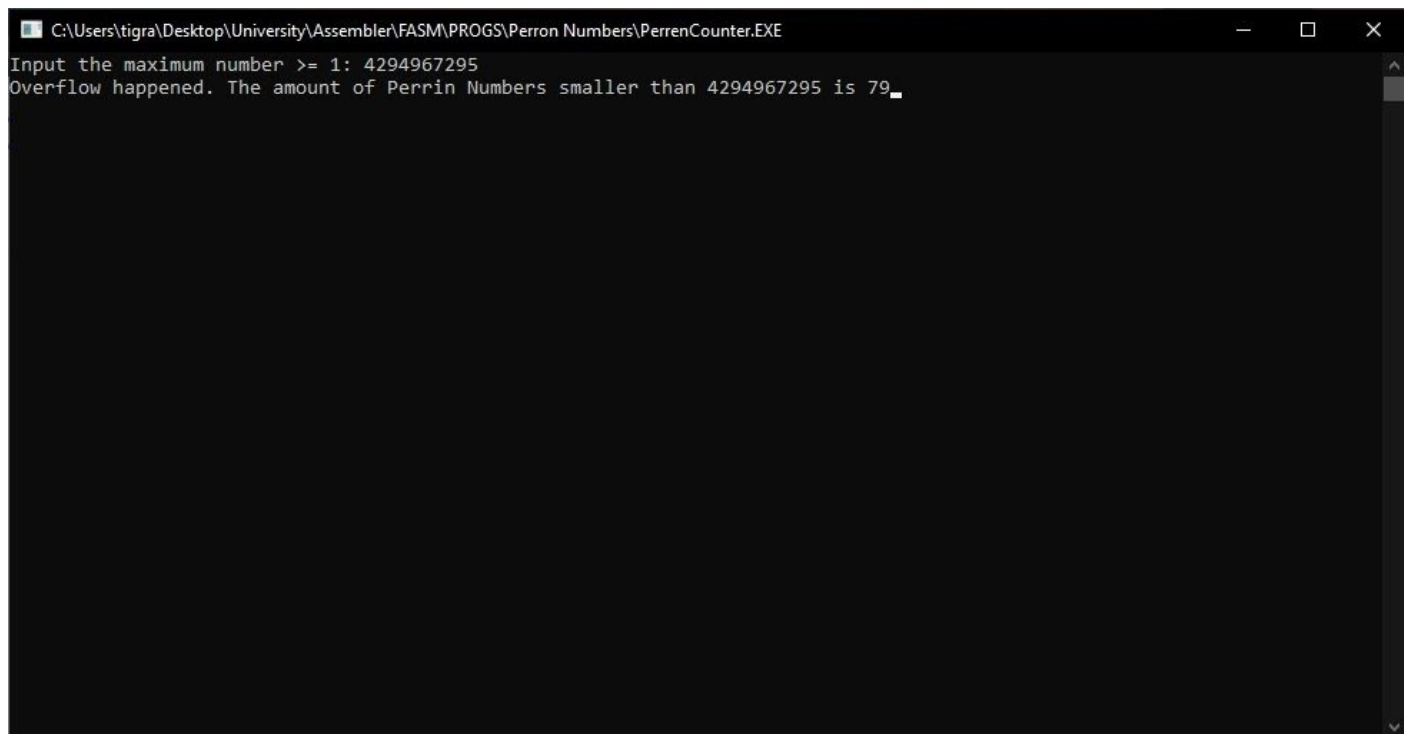
Пример 5. В случае ввода пользователем обычного натурального числа, программа выведет корректное количество чисел Перрена, которые меньше введенного числа.



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\tigra\Desktop\University\Assembler\FASM\PROGS\Perron Numbers\PerrenCounter.EXE. The window has standard minimize, maximize, and close buttons. The command prompt shows the following text: "Input the maximum number >= 1: 5" followed by "Calculations finished. The amount of Perrin Numbers smaller than 5 is 7". The background is black, and the text is white.

```
C:\Users\tigra\Desktop\University\Assembler\FASM\PROGS\Perron Numbers\PerrenCounter.EXE
Input the maximum number >= 1: 5
Calculations finished. The amount of Perrin Numbers smaller than 5 is 7
```

Пример 6. В случае ввода максимального значения $2^{32} - 1$, программа выведет корректное количество чисел Перрена, которые меньше $2^{32} - 1$.



```
C:\Users\tigra\Desktop\University\Assembler\FASM\PROGS\Perron Numbers\PerrenCounter.EXE
Input the maximum number >= 1: 4294967295
Overflow happened. The amount of Perrin Numbers smaller than 4294967295 is 79_
```

Приложение №1.

; Кочарян Тигран, БПИ199, Вариант 10.

; Разработать программу, определяющую число чисел
Перрена

; от 1 до беззнакового машинного слова.

format PE console

entry start

include 'include\win32a.inc'

;-----

; Секция .data с объявленными переменными.

;-----

section '.data' data readable writable

; Создание переменных, которые будут
использованы в дальнейшем.

strInputMaximum db 'Input the maximum number >= 1: ', 0

strIncorrectMaximum db 'Incorrect maximum value = %u', 10, 0

strCheck db 'Current maximum = %u', 10, 0

strFinish db 'Calculations finished. The amount of Perrin Numbers
smaller than %u is %u', 0

strOverflowFinish db 'Overflow happened. The amount of Perrin Numbers
smaller than %u is %u', 0

strScanMax db '%u', 0

; Создание массива из 4 элементов, для хранения
последовательности чисел Перрена

arrayNumbers rd 4

maxNumber dd ?

currentCount dd 1

;-----

; Секция .code для вызова различных процедур.

;-----

section '.code' code readable executable

start:

; 1) Программа запрашивает пользователя ввести
максимальное число от 1 до $(2^{32})-1$

call MaximumInput

; 2) Затем программа заполняет массив
первоначальными элементами 3 2 5 5. Если введенное
maxNumber <= 6,

; то пользователь получит ответ.

call FillInitialCheck

; 3) Программа итерирует процедуру до тех пор, пока
arrayNumbers[3] <= maxNumber.

call IteratePerrins

;-----

; Запрос ввода верхней границы расчетов - DD число.

;-----

MaximumInput:

; Печать сообщения для ввода числа maxNumber.

push strInputMaximum

call [printf]

add esp, 4

push maxNumber

push strScanMax

call [scanf]

add esp, 8

; Сравниваем значение [maxNumber] с 1. Если оно меньше,
то завершаем.

cmp [maxNumber], 1

jae endInputArray

; если maxNumber > 1, начинаем вычисления.

; в противном случае, завершаем.

call WrongInput

endInputArray:

ret

```

;-----
; Заполнение массива первичными значениями
последовательности чисел Перрена
; и делаем первичные проверки для maxNumber <= 6.
;-----
FillInitialCheck:
;    Заполнение массива.
    mov dword[arrayNumbers], 3
    mov dword[arrayNumbers+4], 2
    mov dword[arrayNumbers+8], 5
    mov dword[arrayNumbers+12], 5
;    Начало сравнения maxNumber с 1,2,4,6 и вывод
соответствующих сообщений.
;    Сравнение с 1. Увеличение currentCount на 2.
    cmp [maxNumber], 1
    je Finish
;    Сравнение с 2. Увеличение currentCount на 2.
    call IncreaseCountInitial
    cmp [maxNumber], 2
    je Finish
;    Сравнение с 4. Увеличение currentCount на 2.
    call IncreaseCountInitial
    cmp [maxNumber], 4
    jbe Finish
;    Сравнение с 6. Увеличение currentCount на 2.
    call IncreaseCountInitial
    cmp [maxNumber], 6
    jbe Finish
    ret

;-----
; Итерация по массиву, подсчет значения [arrayNumbers+12]
по формуле:
;  $P(n) = P(n - 2) + P(n - 3)$ .

```

;-----

IteratePerrins:

; также для переноса значения массива мог бы
быть использован

; memmove(arrayNumbers, arrayNumbers+4, 12) for moveing the array.

; но это занимает немного больше времени.
Поэтому был использован такой способ:

; В регистры eax, ebx, ecx переносятся значения
массива, происходит подсчет и сдвиг.

; Затем происходит зачистка регистров после
работы с ними.

mov eax, [arrayNumbers+12]

mov ebx, [arrayNumbers+8]

mov ecx, [arrayNumbers+4]

mov [arrayNumbers+8], eax

mov [arrayNumbers+4], ebx

mov [arrayNumbers], ecx

; Зачистка регистров.

xor eax, eax

xor ebx, ebx

xor ecx, ecx

; Подсчет нового значения для [arrayNumbers+12] по
формуле $P(n) = P(n - 2) + P(n - 3)$.

mov ebx, [arrayNumbers+4]

add ebx, [arrayNumbers]

; Если [arrayNumbers+12] переполняет беззнаковое
двойное слово,

; программа завершается и выводит текущее
значение currentCount.

jc OverflowFinish

mov [arrayNumbers+12], ebx

; Перенос значения [arrayNumbers+12] в регистр ebx для
сравнения.

mov ebx, [arrayNumbers+12]

```

        cmp [maxNumber],ebx
;      Если [maxNumber] < [arrayNumbers+12], то завершаем подсчеты и
выводи ответ.

        jb Finish

;      Увеличиваем значение currentCount на 1 и повторяем
итерацию.

        call IncreaseCount

        jmp IteratePerrins

;-----
; Увеличение счетчика либо на 1, либо на 2
(увеличение на 2 необходимо в случае некоторых
первых чисел чисел,
; которые повторяются 2 раза в последовательности.
;-----

;      Копирование currentCount в регистр eax, увеличение на 2
и копирование из eax в currentCount.
IncreaseCountInitial:

        mov eax, [currentCount]
        add eax, 2
        mov [currentCount], eax
        ret

;      Копирование currentCount в регистр eax, увеличение на 1
и копирование из eax в currentCount.
IncreaseCount:

        mov eax, [currentCount]
        add eax, 1
        mov [currentCount], eax
        ret

;-----
; Завершение программы либо с сообщением об ошибке,
либо с выводом ответа -

```

```
; количества чисел из последовательности Перрена,  
которые меньше maxNumber.
```

```
;-----
```

```
; Вывод сообщения об ошибки с некорректным  
значением maxNumber.
```

WrongInput:

```
; Если ввод некорректный, выводится сообщение.
```

```
push [maxNumber]
```

```
push strIncorrectMaximum
```

```
call [printf]
```

```
; Ожидание нажатия клавиши для закрытия.
```

```
add esp, 8
```

```
call [getch]
```

```
; Завершение.
```

```
push 0
```

```
call [ExitProcess]
```

OverflowFinish:

```
; Вывод текущего значения счетчика с учетом того,
```

```
; что произошло переполнение беззнакового  
двойного слова.
```

```
push [currentCount]
```

```
push [maxNumber]
```

```
push strOverflowFinish
```

```
call [printf]
```

```
; Ожидание нажатия клавиши для закрытия.
```

```
add esp, 12
```

```
call [getch]
```

```
; Завершение.
```

```
push 0
```

```
call [ExitProcess]
```

```
; Вывод текущего значения счетчика в консоль и  
завершение.
```


Finish:

```
;    Вывод значения счетчика.  
    push [currentCount]  
    push [maxNumber]  
    push strFinish  
    call [printf]  
;  
    Ожидание нажатия клавиши для закрытия.  
    add esp, 12  
    call [getch]  
;  
    Завершение.  
    push 0  
    call [ExitProcess]
```

```

;-----
; И м п о р т и р о в а н н ы е   б и б л и о т е к и   в   с е к ц и и   .data: user32.inc,
kernel32.inc и т.д.
;-----

section '.idata' import data readable

    library kernel, 'kernel32.dll',\
        msvcrt, 'msvcrt.dll',\
        user32, 'USER32.DLL'

include 'include\api\user32.inc'
include 'include\api\kernel32.inc'

    import kernel,\
        ExitProcess, 'ExitProcess',\
        HeapCreate, 'HeapCreate',\
        HeapAlloc, 'HeapAlloc'
include 'include\api\kernel32.inc'

    import msvcrt,\
        printf, 'printf',\
        memmove, 'memmove',\
        scanf, 'scanf',\
        getch, '_getch'

```