

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

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

**МИКРОПРОЕКТ № 1
ПО ДИСЦИПЛИНЕ
«АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ»**

<i>Подп. и дата</i>	
<i>Инв. №</i>	
<i>Взам. инв.</i>	
<i>Подп. и дата</i>	
<i>Инв. №</i>	

**Пояснительная записка
ЛИСТ УТВЕРЖДЕНИЯ**

RU.17701729.04.13-01 81 01-1-ЛУ

Исполнитель
студент группы БПИ191
Н.К. Игумнов

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ	3
1. Наименование темы работы. Текст задания.....	3
2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ	4
2.1. Описание алгоритма и функционирования программы.....	4
2.1.1. Соглашения, принятые при составлении псевдокода	4
2.1.1.1. Описание входных данных.....	5
2.1.1.2. Описание выходных данных	5
2.1.1.3. Описание алгоритма	6
2.1.1.3.1. Временная сложность алгоритма	6
3. ТЕСТИРОВАНИЕ ПРОГРАММЫ.....	7
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	11
ПРИЛОЖЕНИЕ 1	12
ПРИЛОЖЕНИЕ 2	13
ПРИЛОЖЕНИЕ 3	14

1. ВВЕДЕНИЕ

1. Наименование темы работы. Текст задания

Требуется разработать программу на NASM, определяющую максимальное значение параметра функции факториала, при котором значение функции не выходит за пределы положительного целого размером в двойное машинное слово.

2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

2.1. Описание алгоритма и функционирования программы

2.1.1. Соглашения, принятые при составлении псевдокода

При составлении псевдокода используются следующие соглашения (очень похожи на соглашения, принятые в «Алгоритмы: построение и анализ» [1, с. 42-44]).

- 1) Блочная структура указывается с помощью отступов. Этот способ применения отступов применяется для цикла **do-while**.
- 2) Конструкция цикла **do-while** интерпретируется, как в языках программирования C, C++, C#, Java.
- 3) Символ `“//”` указывает, что остальная часть строки представляет собой комментарий.
- 4) Переменные (*prev*, *n* и *next*) являются локальными для данной процедуры. Я не буду использовать глобальные переменные без явного указания этого факта
- 5) Присваивание одной переменной другой происходит по значению (с копированием исходных данных)
- 6) Инструкция **return** немедленно возвращает управление в точку вызова в вызывающей процедуре. В большинстве случаев инструкция **return** получает значение для возврата вызывающей процедуре.
- 7) Булевы операторы “и” и “или” вычисляются *сокращенно* (short circuiting). Это означает, что при вычислении выражения “*x* и *y*” сначала вычисляется значение выражения *x*. Если это значение ложно (*false*), то всё выражение не может быть истинным, и значение выражения *y* не вычисляется. Если же выражение *x* истинно (*true*), то для определения значения всего выражения необходимо вычислить выражение *y*. Аналогично в выражении “*x* или *y*” величина *y* вычисляется только в том случае, если выражение *x* ложно.

2.1.1.1. Описание входных данных

На вход алгоритму поступает беззнаковое целое положительное число N размера двойного слова, введённое пользователем с консоли. То есть $N \in [1, 2^{32} - 1]$, $N \in \mathbb{N}$, (так как $2^{32} - 1$ – максимальное значение беззнакового двойного слова).

2.1.1.2. Описание выходных данных

Выходным параметром является положительное целое число в промежутке $[1, 12]$.
Так как $13! > 2^{32} - 1$.

2.1.1.3. Описание алгоритма

Из определения факториала следует, что $n! = (n - 1)! \cdot n$. Пройдёмся циклом с постусловием по всем факториалам, размер которых – двойное слово.

Если текущее значение больше введённого числа, то цикл завершается.

Как выяснить, что текущее значение факториала не поместилось в данный размер (то есть оно больше максимального значения беззнакового двойного слова, а следовательно, превышает введённое число)? При переполнении будет взято число по модулю $2^{32} - 1$. Из определения $(n - 1)! = n! / n$ ($n > 0$). Если происходит переполнение, то это равенство выполняться не будет и цикл завершается.

После завершения цикла на экран выведется предыдущий параметр факториала (который удовлетворял обоим условиям).

Ниже приведён псевдокод алгоритма:

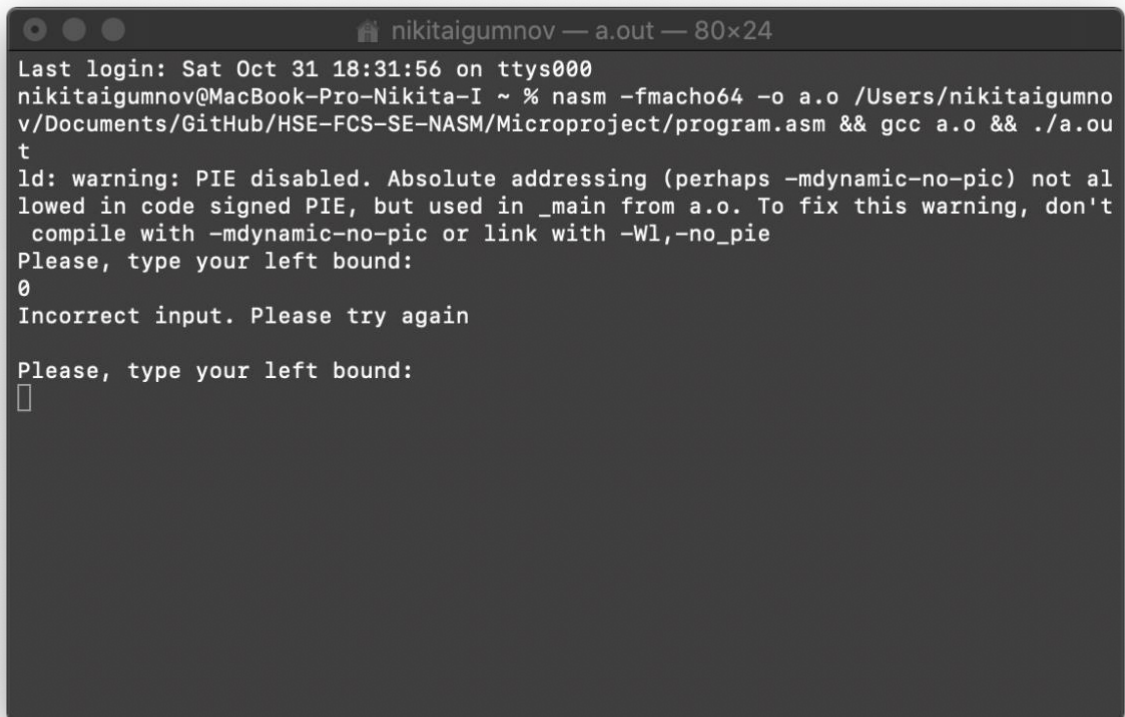
```
getAns(N)
1  prev = 1 // предыдущее значение факториала
2  n = 1 // текущий параметр факториала
3  next = 1 // текущее значение факториала
4  do
5      n += 1 // увеличиваем параметр факториала на 1
6      prev = next // новое "предыдущее значение"
7      next *= n // новое "текущее значение"
8  while (prev == next / n и next ≤ N)
9  n -= 1
10 return n
```

2.1.1.3.1. Временная сложность алгоритма

Алгоритм работает за $O(\log N)$, где N – входное число. Пусть $n : N \leq n!$, причём n – минимально (> 0). Тогда временная сложность – $O(n)$. Заметим, что $\log N \geq \log n!$. По формуле Стирлинга, $\log n! = n \log n - n \log e + O(\log n)$. Получаем, что $\log N \geq n \log n - n \log e + O(\log n) \geq n$ при $n \geq 3$.

3. ТЕСТИРОВАНИЕ ПРОГРАММЫ

1. Некорректный ввод ($N = 0$)

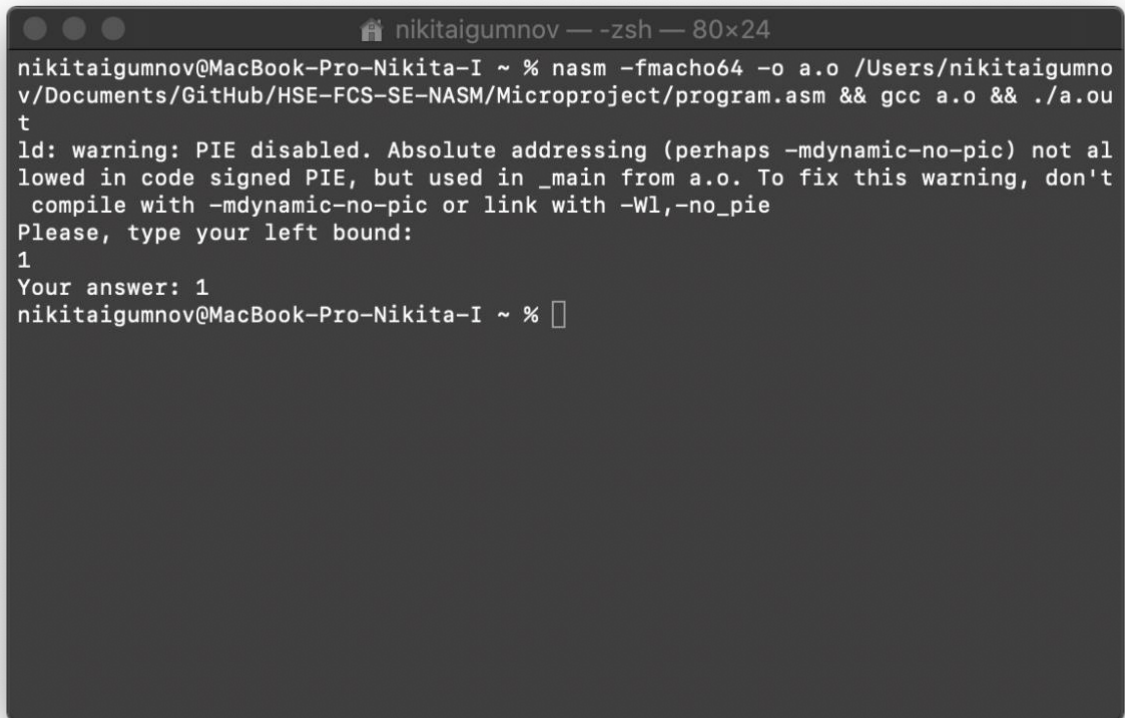


```
nikitaigumnov — a.out — 80x24
Last login: Sat Oct 31 18:31:56 on ttys000
nikitaigumnov@MacBook-Pro-Nikita-I ~ % nasm -fmacho64 -o a.o /Users/nikitaigumnov/Documents/GitHub/HSE-FCS-SE-NASM/Microproject/program.asm && gcc a.o && ./a.out
ld: warning: PIE disabled. Absolute addressing (perhaps -mdynamic-no-pic) not allowed in code signed PIE, but used in _main from a.o. To fix this warning, don't compile with -mdynamic-no-pic or link with -Wl,-no_pie
Please, type your left bound:
0
Incorrect input. Please try again

Please, type your left bound:
█
```

Программа отработала успешно.

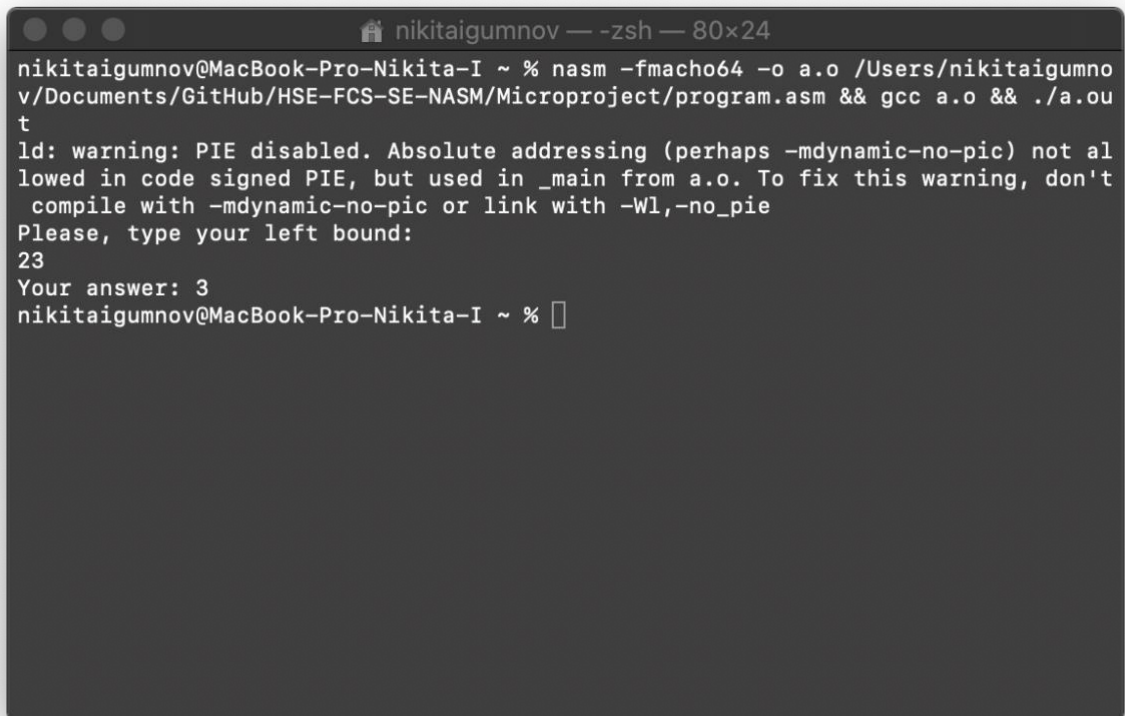
2. $N = 1$



```
nikitaigumnov@MacBook-Pro-Nikita-I ~ % nasm -fmacho64 -o a.o /Users/nikitaigumnov/Documents/GitHub/HSE-FCS-SE-NASM/Microproject/program.asm && gcc a.o && ./a.out
ld: warning: PIE disabled. Absolute addressing (perhaps -mdynamic-no-pic) not allowed in code signed PIE, but used in _main from a.o. To fix this warning, don't compile with -mdynamic-no-pic or link with -Wl,-no_pie
Please, type your left bound:
1
Your answer: 1
nikitaigumnov@MacBook-Pro-Nikita-I ~ %
```

Программа отработала успешно.

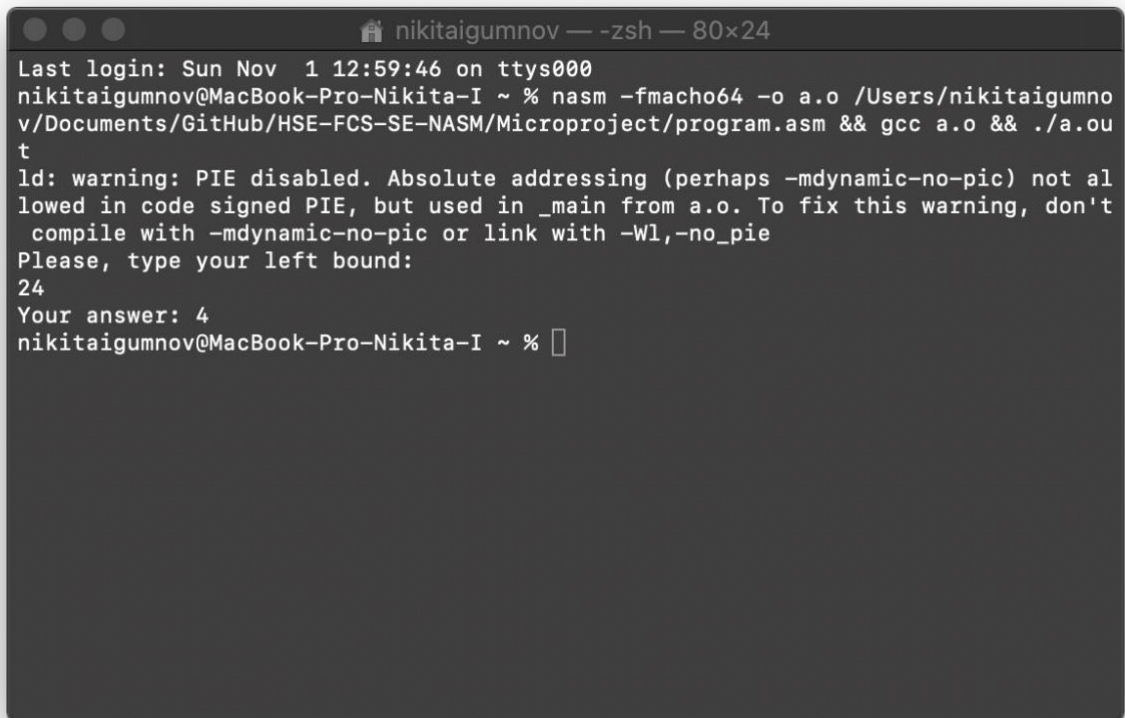
3. $N = 4! - 1 = 23$ – пограничные числа



```
nikitaigumnov@MacBook-Pro-Nikita-I ~ % nasm -fmacho64 -o a.o /Users/nikitaigumnov/Documents/GitHub/HSE-FCS-SE-NASM/Microproject/program.asm && gcc a.o && ./a.out
ld: warning: PIE disabled. Absolute addressing (perhaps -mdynamic-no-pic) not allowed in code signed PIE, but used in _main from a.o. To fix this warning, don't compile with -mdynamic-no-pic or link with -Wl,-no_pie
Please, type your left bound:
23
Your answer: 3
nikitaigumnov@MacBook-Pro-Nikita-I ~ %
```

Программа отработала успешно.

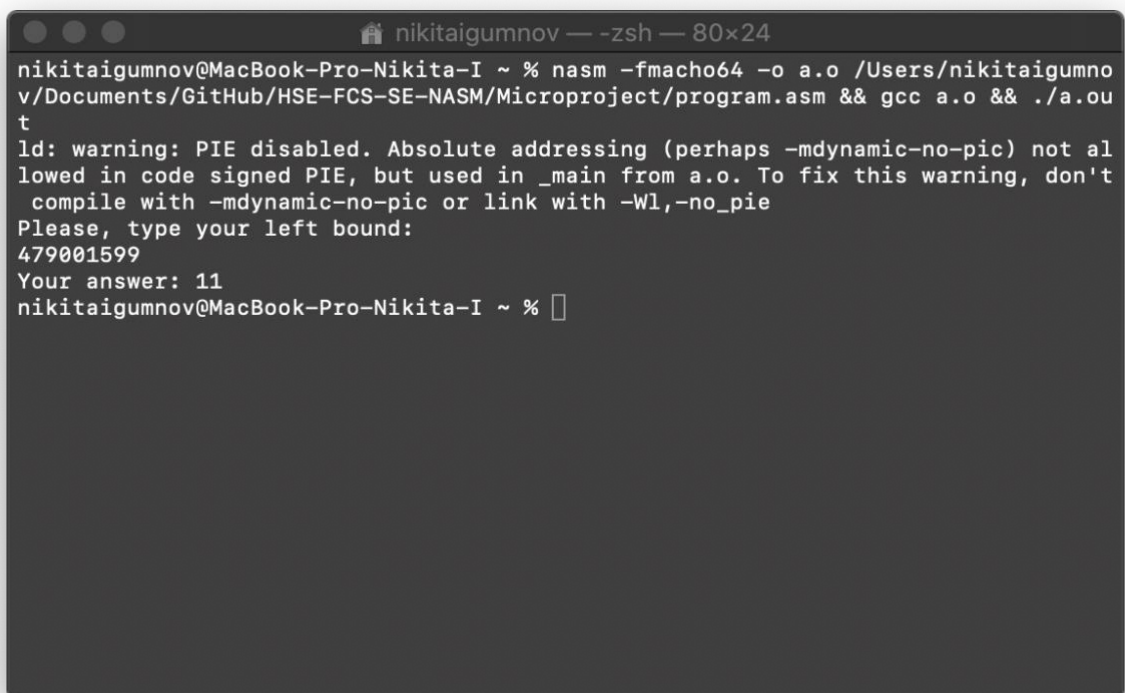
4. $N = 4! = 24$ – пограничные числа



```
nikitaigumnov — zsh — 80x24
Last login: Sun Nov  1 12:59:46 on ttys000
nikitaigumnov@MacBook-Pro-Nikita-I ~ % nasm -fmacho64 -o a.o /Users/nikitaigumnov/Documents/GitHub/HSE-FCS-SE-NASM/Microproject/program.asm && gcc a.o && ./a.out
ld: warning: PIE disabled. Absolute addressing (perhaps -mdynamic-no-pic) not allowed in code signed PIE, but used in _main from a.o. To fix this warning, don't compile with -mdynamic-no-pic or link with -Wl,-no_pie
Please, type your left bound:
24
Your answer: 4
nikitaigumnov@MacBook-Pro-Nikita-I ~ %
```

Программа отработала успешно.

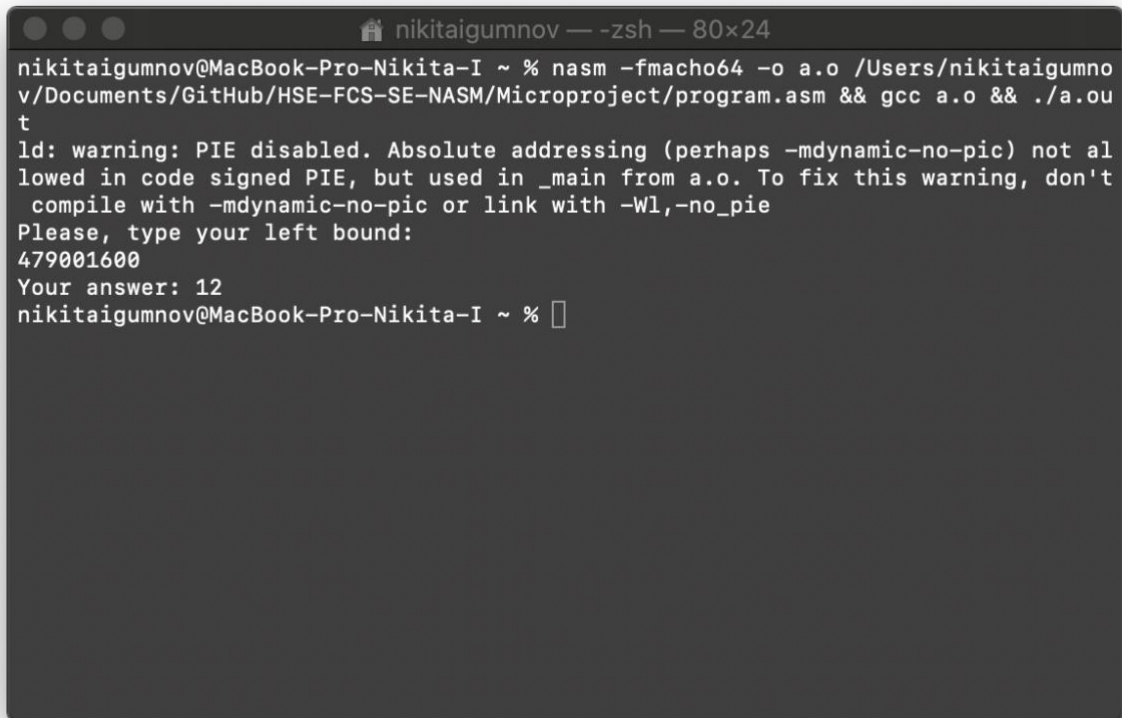
5. $N = 12! - 1 = 479001599$ – пограничные числа



```
nikitaigumnov — zsh — 80x24
nikitaigumnov@MacBook-Pro-Nikita-I ~ % nasm -fmacho64 -o a.o /Users/nikitaigumnov/Documents/GitHub/HSE-FCS-SE-NASM/Microproject/program.asm && gcc a.o && ./a.out
ld: warning: PIE disabled. Absolute addressing (perhaps -mdynamic-no-pic) not allowed in code signed PIE, but used in _main from a.o. To fix this warning, don't compile with -mdynamic-no-pic or link with -Wl,-no_pie
Please, type your left bound:
479001599
Your answer: 11
nikitaigumnov@MacBook-Pro-Nikita-I ~ %
```

Программа отработала успешно

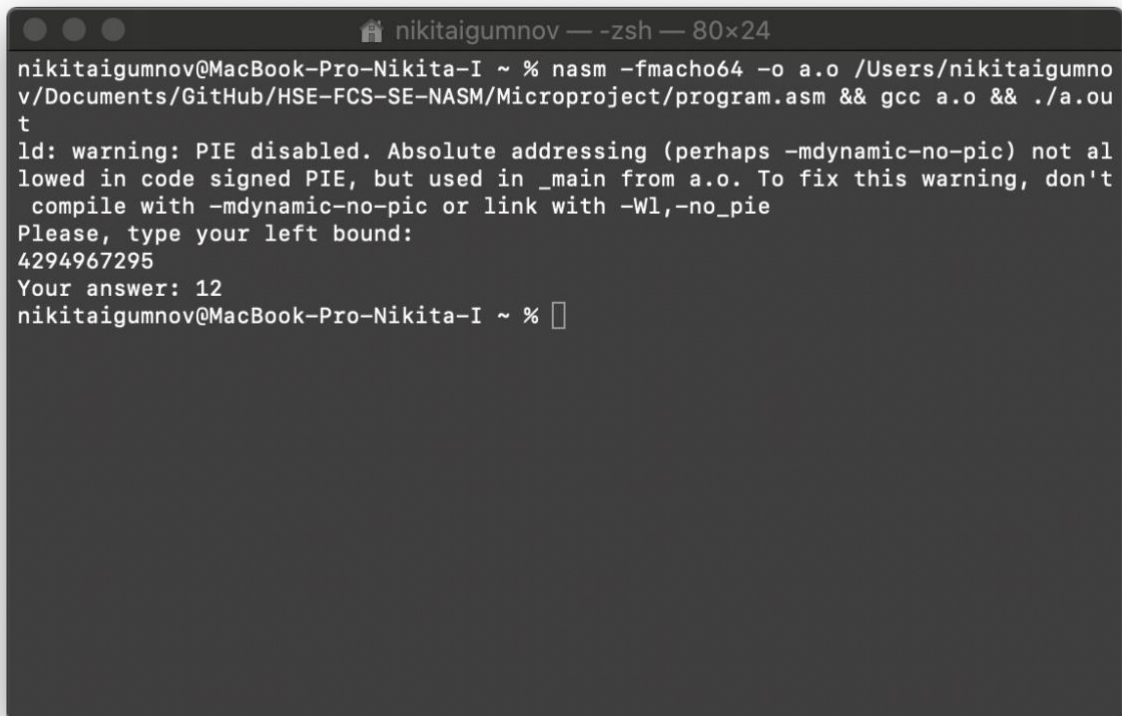
6. $N = 12! = 479001600$ – пограничные числа



```
nikitaigumnov@MacBook-Pro-Nikita-I ~ % nasm -f macho64 -o a.o /Users/nikitaigumnov/Documents/GitHub/HSE-FCS-SE-NASM/Microproject/program.asm && gcc a.o && ./a.out
ld: warning: PIE disabled. Absolute addressing (perhaps -mdynamic-no-pic) not allowed in code signed PIE, but used in _main from a.o. To fix this warning, don't compile with -mdynamic-no-pic or link with -Wl,-no_pie
Please, type your left bound:
479001600
Your answer: 12
nikitaigumnov@MacBook-Pro-Nikita-I ~ %
```

Программа отработала успешно

7. $N = 2^{32} - 1 = 4294967295$ – работа с переполнением



```
nikitaigumnov@MacBook-Pro-Nikita-I ~ % nasm -f macho64 -o a.o /Users/nikitaigumnov/Documents/GitHub/HSE-FCS-SE-NASM/Microproject/program.asm && gcc a.o && ./a.out
ld: warning: PIE disabled. Absolute addressing (perhaps -mdynamic-no-pic) not allowed in code signed PIE, but used in _main from a.o. To fix this warning, don't compile with -mdynamic-no-pic or link with -Wl,-no_pie
Please, type your left bound:
4294967295
Your answer: 12
nikitaigumnov@MacBook-Pro-Nikita-I ~ %
```

Программа отработала успешно

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Алгоритмы: построение и анализ, 3-е изд. / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн; Пер. с англ. – СПб.: ООО “Диалектика”, 2019. – 1328 С.
- 2) x86_64 NASM Assembly Quick Reference ("Cheat Sheet") [Электронный ресурс]. Режим доступа: https://www.cs.uaf.edu/2017/fall/cs301/reference/x86_64.html, свободный. (дата обращения: 30.10.2020)
- 3) Программирование на языке ассемблера NASM для ОС UNIX / А. В. Столяров – МАКС Пресс, 2011. – 188 С.
- 4) Команды LOOP, LOOPD, LOOPE, LOOPNE, LOOPNZ, LOOPZ [Электронный ресурс]. Режим доступа: <http://sysprog.ru/post/41>, свободный. (дата обращения: 30.10.2020)

ПРИЛОЖЕНИЕ 1

ТЕРМИНОЛОГИЯ

Факториал – функция, определённая на множестве неотрицательных целых чисел. Она определяется как произведение всех натуральных чисел от 1 до n включительно:

$$n! = 1 \cdot 2 \cdot \dots \cdot n = \prod_{k=1}^n k$$

Псевдокод – компактный неформальный язык описания принципа работы алгоритмов, использующий ключевые слова обычных языков программирования, но опускающий несущественные для понимания алгоритма подробности и специфический синтаксис.

ПРИЛОЖЕНИЕ 2

ОПИСАНИЕ ПРОЦЕДУР

Таблица 1 – Описание основных процедур приложения

Процедура	Входные данные	Выходные данные	Назначение
ComputeAnswer	N – входной параметр	n – искомый ответ	Процедура, подсчитывающая искомое значение (смотрите описание алгоритма, п. 2.1.1.3)
WrongInput	-	-	Процедура, вызываемая в случае задания некорректных параметров
ExitProgram	-	-	Процедура для завершения программы

ПРИЛОЖЕНИЕ 3

КОД ПРОГРАММЫ

```
; Эквивалентный код на C
; #include <stdio.h>
; int main(int argc, char *argv[])
; {
;     unsigned int N;
;     printf("Please, type your left bound:\n\r");
;     scanf("%u", &N);
;     if (N < 1) {
;         printf("Incorrect input. Please try again\n\r\n\r");
;         return main(argc, argv);
;     }
;     unsigned int prev = 1, n = 1, next = 1;
;     do {
;         ++n;
;         prev = next;
;         next *= n;
;     } while (prev == next / n && next <= N);
;     --n;
;     printf("Your answer: %u", n);
;     return 0;
; }

; ВАРИАНТ 15

; nasm -f macho64 -o a.o ПУТЬ_ДО_ФАЙЛА && gcc a.o && ./a.out
; nasm -f macho64 -o a.o /Users/nikitaigumnov/Documents/GitHub/HSE-FCS-SE-NASM/Microproject/program.asm
&& gcc a.o && ./a.out

    global  _main
    extern  _scanf
    extern  _printf

section .data

    formatUInt db '%u', 0
    typeBound db 'Please, type your left bound: ', 10, 13, 0
    printAnswer db 'Your answer: %u', 10, 13, 0
    wrongInput db 'Incorrect input. Please try again', 10, 13, 10, 13, 0
```

```
N: dd 0          ; правая граница
n: dd 1          ; текущий параметр факториала
prev: dd 1       ; предыдущее значение факториала – (n - 1)!
next: dd 1       ; текущее значение факториала – n!
```

section .text

```
_main:
    push rbp

    mov rdi, typeBound
    call _printf
    mov rdi, formatUInt
    mov rsi, N
    call _scanf          ; считываем правую границу

    mov eax, [rel N]
    cmp eax, 1
    jb WrongInput       ; когда N < 1 (N = 0)

    mov ecx, 2
    call ComputeAnswer   ; вычисляем ответ

    mov rdi, printAnswer
    mov rsi, [rel n]
    call _printf         ; выводим ответ

    jmp ExitProgram
```

; == ПРОЦЕДУРЫ ==

ComputeAnswer:

```
    mov eax, [rel n]
    inc eax
    mov [rel n], eax      ; ++n

    mov eax, [rel next]
    mov [rel prev], eax   ; prev = next

    mov eax, [rel next]
    mul dword [rel n]
```

```
mov [rel next], eax      ; next *= n

mov eax, [rel next]
mov edx, 0
div dword [rel n]
cmp eax, [rel prev]
jne FinishLoop          ; prev != next / n

mov eax, [rel next]
cmp eax, [rel N]
ja FinishLoop           ; next > N

inc ecx                 ; бесконечный цикл (чтобы можно было выйти, используя "break")
loop ComputeAnswer      ; повторный вызов цикла

FinishLoop:
    mov eax, [rel n]
    dec eax
    mov [rel n], eax     ; --n

ret

WrongInput:
    mov rdi, wrongInput
    call _printf          ; некорректный ввод

    pop rbp
    jmp _main

ExitProgram:
    pop rbp
    mov rax, 0            ; normal, no error, return value
    ret
```