**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Образовательная программа бакалавриата «Программная инженерия»

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. инв. № |  | | Подп. и дата |  | | Инв. № подл. |  | | **Архитектура вычислительных систем**  **Микропроект №1, вариант 4**  **ПРОГРАММА ДЛЯ ОПРЕДЕЛЕНИЯ ВЗАИМНОЙ ПРОСТОТЫ ЧИСЕЛ**  **Пояснительная записка** | | |
|  |  | |
| Исполнитель:  студент группы БПИ191  В.И. Беловицкий/  2020 г. | |
|  | | |
|  | |  |

**СОДЕРЖАНИЕ**

[1. ПОСТАВЛЕННАЯ ЗАДАЧА 3](#_Toc55165168)

[1.1. Задание 3](#_Toc55165169)

[1.2. Дополнительный функционал 3](#_Toc55165170)

[2. ОБЩИЙ АЛГОРИТМ РАБОТЫ 3](#_Toc55165171)

[3. ОПИСАНИЕ ДАННЫХ 3](#_Toc55165172)

[4. ВХОДНЫЕ ДАННЫЕ 4](#_Toc55165173)

[4.1. Ввод количества чисел 4](#_Toc55165174)

[4.2. Последовательный ввод чисел 4](#_Toc55165175)

[5. ИСПОЛЬЗОВАННЫЕ АЛГОРИТМЫ 5](#_Toc55165176)

[6. ТЕСТИРОВАНИЕ 5](#_Toc55165177)

[7. ТЕКСТ ПРОГРАММЫ 6](#_Toc55165178)

1. ПОСТАВЛЕННАЯ ЗАДАЧА
   1. Задание

Разработать программу, решающую вопрос - являются ли четыре заданных числа взаимно простыми (числа задать машинными словами без знака).

* 1. Дополнительный функционал

Была добавлена возможность вводить количество чисел. Доступный диапазон ограничен.

1. ОБЩИЙ АЛГОРИТМ РАБОТЫ

Программа выполняется по следующему алгоритму:

* Ввод количества чисел
* Повтор ввода при некорректном вводе
* Последовательный ввод чисел
* Повтор ввода при некорректном вводе
* Последовательное выполнение алгоритма Евклида для введенных чисел
* Вывод результата – НОД чисел + являются ли они взаимно простыми

1. ОПИСАНИЕ ДАННЫХ

В приведенной ниже таблицы указаны переменные

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Описание | Тип | Константное значение |
| vec\_size | вводимый размер массива (количество чисел) | dd | - |
| vec\_min\_size | минимальный размер массива | dd | 1 |
| vec\_max\_size | максимальный размер | dd | 10 |
| num\_min | минимальное значение введенного числа | dd | 1 |
| num\_max | максимальное значение введенного числа | dd | 65535 |
| i | временный индекс массива | dd | - |
| tmp | временная переменная для хранения значения в регистре | dd | - |
| vec | ссылка на первый элемент массива (массив последовательно располагаться в ячейках, следующих за первой) | dd | - |

1. ВХОДНЫЕ ДАННЫЕ
   1. Ввод количества чисел

При запуске программы пользователю предлагается ввести количество чисел, которые будут проверяться на взаимную простоту.

**Все операции вывода и ввода в программе интерпретируют числа как 16-битное число без знака.**

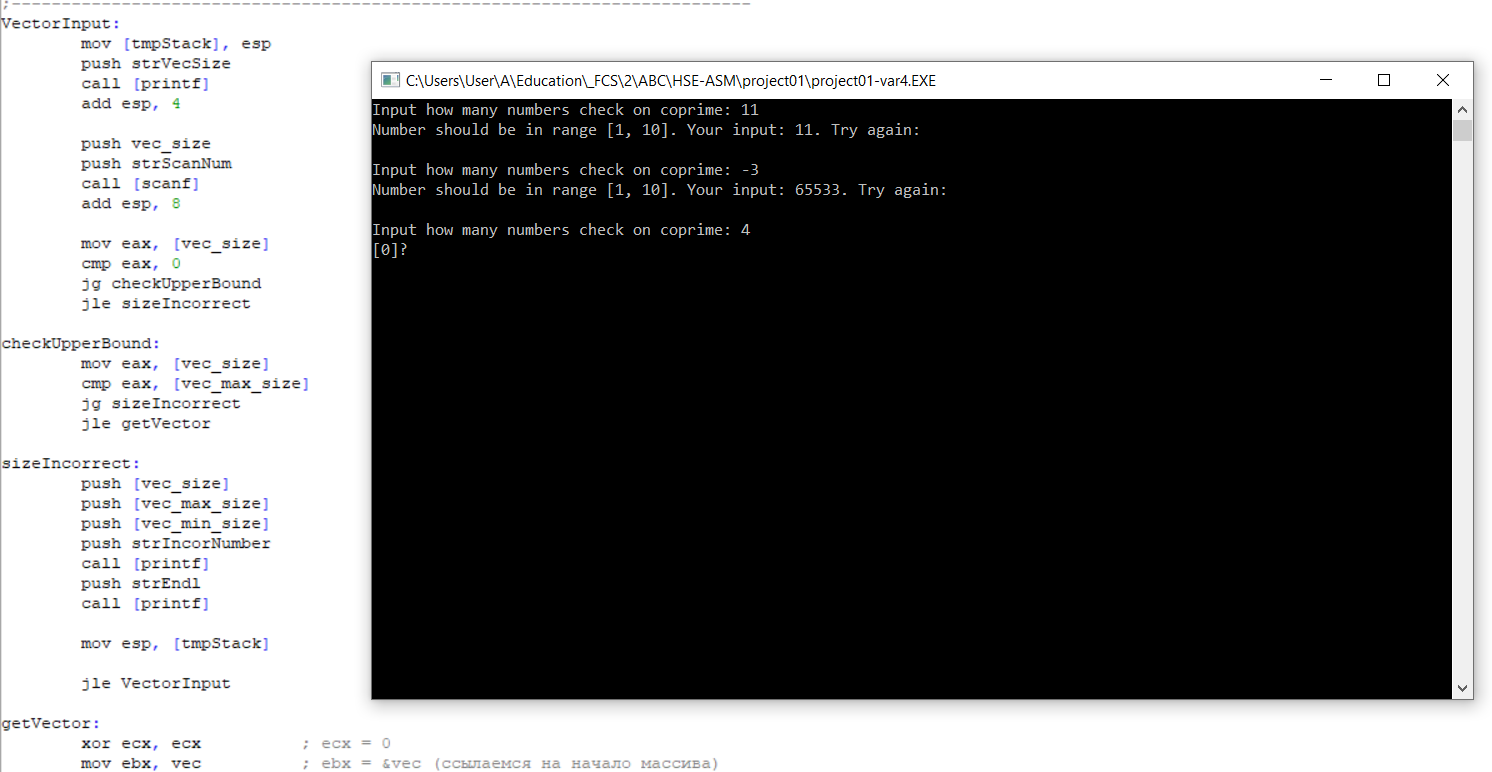


Рис 1. Пример ввода количества чисел.

* 1. Последовательный ввод чисел

Числа вводятся последовательно, осуществляется проверка в соответствии с заданными num\_min и num\_max (был намеренно исключен 0). Поскольку числа всегда интерпретируются как как 16-битные число без знака, некорректным считается только 0 (искать для него НОД не имеет смысла).

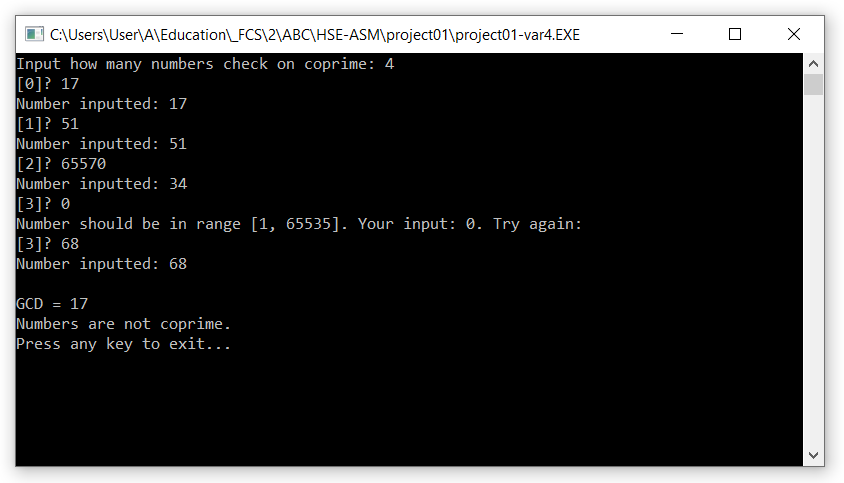


Рис. 2. Пример ввода последовательности чисел.

1. ИСПОЛЬЗОВАННЫЕ АЛГОРИТМЫ

Взаимно простые числа – числа, наибольший общий делитель (НОД) которых равен 1.

Для поиска НОД использовался алгоритм Евклида, использующий вычитание большего числа из меньшего. Алгоритм описан ниже:

* Первый элемент массива записывается в eax. На последующих итерациях в eax лежит НОД предыдущей пары элементов.
* Каждый последующий элемент записывается в edx
* Если edx>eax, значения в регистрах меняем местами, в eax должно быть всегда наибольшее значение.
* Из eax вычитается edx, пока edx – не станет равным 0. В eax остается НОД пары чисел.
* Если дошли до конца массива – выходим из цикла, в eax лежит НОД множества чисел.

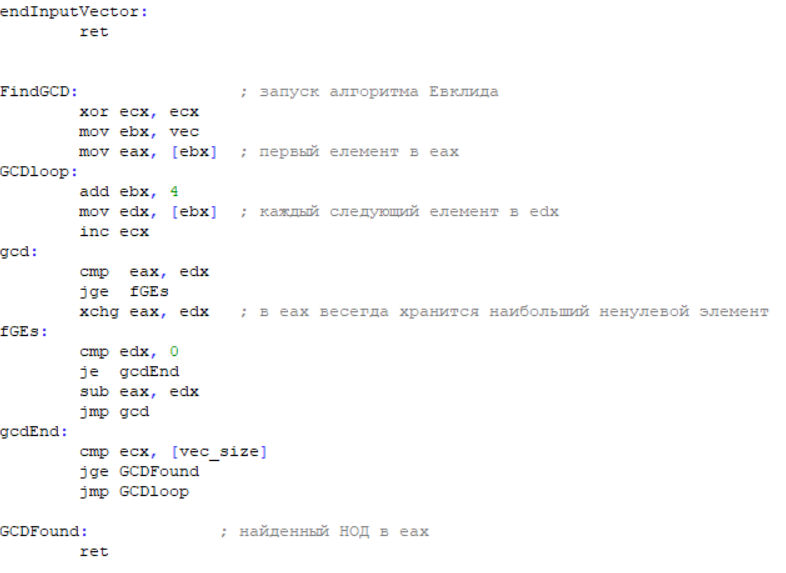
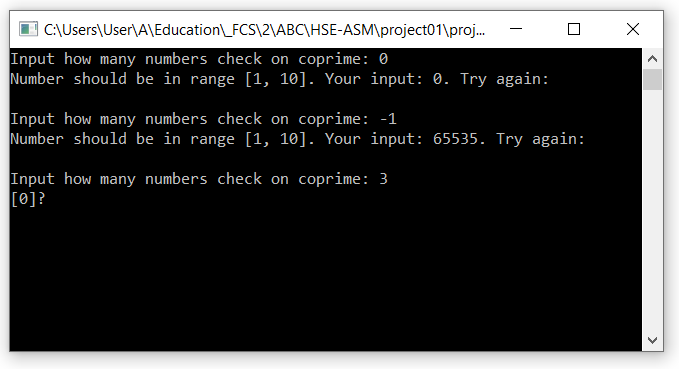


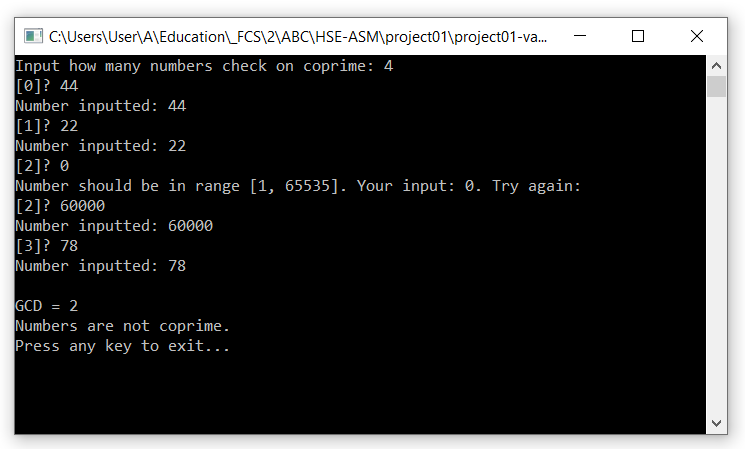
Рис. 3. Алгоритм Евклида для множества чисел

1. ТЕСТИРОВАНИЕ
   1. Тест ввода количества чисел



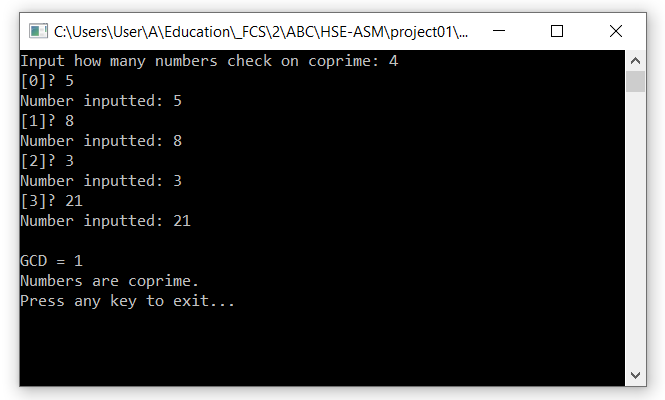
* 1. Тест ввода последовательности чисел

Введем несколько чисел, делящихся на 2. Они не взаимно простые, НОД != 1.



* 1. Тест на взаимною простоту

Числа являются взаимно простыми, НОД = 1.



1. ТЕКСТ ПРОГРАММЫ

format PE console

entry start

include 'win32a.inc'

section '.data' data readable writable

strVecSize db 'Input how many numbers check on coprime: ', 0

strIncorNumber db 'Number should be in range [%hu, %hu]. Your input: %hu. Try again:',10, 0

strValidNumber db 'Number inputted: %hu',10,0

strVecElemI db '[%hu]? ', 0

strScanNum db '%hu', 0

strNotCoprime db 'Numbers are not coprime.', 0

strAreCoprime db 'Numbers are coprime.', 0

strGCD db 'GCD = %hu', 0

strEndl db '', 10, 0

strPressKey db 'Press any key to exit...', 10, 0

vec\_size dd 0

vec\_min\_size dd 1

vec\_max\_size dd 10

num\_min dd 1

num\_max dd 65535

i dd ?

tmp dd ?

tmpStack dd ?

vec dd 0

;--------------------------------------------------------------------------

section '.code' code readable executable

start:

call VectorInput

push strEndl

call [printf]

call FindGCD

mov [tmp], eax

push eax

push strGCD

call [printf]

push strEndl

call [printf]

cmp [tmp], 1

je AreCoprime

push strNotCoprime

call [printf]

jmp finish

AreCoprime:

push strAreCoprime

call [printf]

finish:

push strEndl

call [printf]

push strPressKey

call [printf]

call [getch]

push 0

call [ExitProcess]

;--------------------------------------------------------------------------

VectorInput:

mov [tmpStack], esp

push strVecSize

call [printf]

add esp, 4

push vec\_size

push strScanNum

call [scanf]

add esp, 8

mov eax, [vec\_size]

cmp eax, 0

jg checkUpperBound

jle sizeIncorrect

checkUpperBound:

mov eax, [vec\_size]

cmp eax, [vec\_max\_size]

jg sizeIncorrect

jle getVector

sizeIncorrect:

push [vec\_size]

push [vec\_max\_size]

push [vec\_min\_size]

push strIncorNumber

call [printf]

push strEndl

call [printf]

mov esp, [tmpStack]

jle VectorInput

getVector:

xor ecx, ecx

mov ebx, vec

getVecLoop:

mov [tmp], ebx

cmp ecx, [vec\_size]

jge endInputVector

inputNumber:

mov [i], ecx

push ecx

push strVecElemI

call [printf]

add esp, 8

push ebx

push strScanNum

call [scanf]

add esp, 8

mov eax, [ebx]

cmp eax, [num\_max]

jg incorNumber

cmp eax, [num\_min]

jl incorNumber

jmp correctNumber

incorNumber:

push eax

push [num\_max]

push [num\_min]

push strIncorNumber

call [printf]

add esp, 16

mov ecx, [i]

mov ebx, [tmp]

jmp inputNumber

correctNumber:

mov eax, [ebx]

push eax

push strValidNumber

call [printf]

add esp, 8

mov ecx, [i]

inc ecx

mov ebx, [tmp]

add ebx, 4

jmp getVecLoop

endInputVector:

ret

FindGCD:

xor ecx, ecx

mov ebx, vec

mov eax, [ebx]

GCDloop:

add ebx, 4

mov edx, [ebx]

inc ecx

gcd:

cmp eax, edx

jge fGEs

xchg eax, edx

fGEs:

cmp edx, 0

je gcdEnd

sub eax, edx

jmp gcd

gcdEnd:

cmp ecx, [vec\_size]

jge GCDFound

jmp GCDloop

GCDFound:

ret

;--------------------------------------------------------------------------

section '.idata' import data readable

library kernel, 'kernel32.dll',\

msvcrt, 'msvcrt.dll',\

user32, 'USER32.DLL'

include 'api\user32.inc'

include 'api\kernel32.inc'

import kernel,\

ExitProcess, 'ExitProcess'

include 'api\kernel32.inc'

import msvcrt,\

printf, 'printf',\

scanf, 'scanf',\

getch, '\_getch'