

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА ДЛЯ МИКРОПРОЕКТА №1

КОРИЦКОГО АЛЕКСАНДРА

ФКН БПИ196

ВАРИАНТ 2

ФОРМУЛИРОВКА ЗАДАНИЯ

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

ЛИСТИНГ ПРОГРАММЫ

```
format PE console
include 'win32a.inc'
entry start

section 'data' data readable writeable

str1 db 'Enter 4 numbers: ',0 ;строки для вывода в консоль
str2 db 'Numbers are prime',0
str3 db 'Numbers are not prime',0
str4 db 'Incorrect input',0
strGetInt db '%d', 0 ;строка для scanf
var1 dd ? ;переменные для 4х чисел
var2 dd ?
var3 dd ?
var4 dd ?
prime dd 1 ;переменная для ответа

section 'text' code executable readable
start:
push var1;ввод первой переменной
push strGetInt
call [scanf]
test eax,eax;проверка успешности ввода
jz .error
push var2;ввод второй переменной
push strGetInt
call [scanf]
test eax,eax
jz .error
push var3;ввод третьей переменной
push strGetInt
call [scanf]
test eax,eax
jz .error
push var4;ввод четвертой переменной
push strGetInt
```

```
call [scanf]
test eax,eax
jz .error
call CheckPrime;вызов функции проверки простоты чисел
cmp [prime],2 ;проверка на наличие ошибки
je .error
```

```
cmp [prime],1;если prime==1 то вывести str2(Numbers are prime) иначе
str3(Numbers are not prime)
je .yes
push str3
call [printf]
jmp .end ;пропуск вывода 2й строки
.yes:
push str2
call [printf]
.end:
```

```
call [getch];ожидание
push 0
call [ExitProcess];завершение программы
.error:
push str4
call [printf]
jmp .end
```

```
CheckPrime:
mov eax,[var1] ;поиск минимального числа (min=var1)
cmp eax,[var2] ;если min>var2 то min=var2
jl .m1
mov eax,[var2]
.m1:
cmp eax,[var3] ;если min>var3 то min=var3
jl .m2
mov eax,[var3]
.m2:
cmp eax,[var4] ;если min>var4 то min=var4
jl .m3
mov eax,[var4]
.m3:
```

```
cmp eax,1;проверка правильности ввода, чтобы все числа были больше нуля
jae .ok
mov [prime],2
jmp .end
```

.ok:

mov ecx,eax ;перекладываем минимальное число в ecx
.loop::начало цикла
mov eax,[var1];копируем первое число в eax
xor edx,edx;подготавливаемся к делению, очищаем edx
div ecx;делим на текущее число
test edx,edx;проверяем остаток от деления на равенство нулю
jnz .skip;если остаток не равен нулю то это число не подходит, переходим к
концу цикла

mov eax,[var2];повторяем, берем остатки от деления от других трех чисел
xor edx,edx
div ecx
test edx,edx
jnz .skip

mov eax,[var3]
xor edx,edx
div ecx
test edx,edx
jnz .skip

mov eax,[var4]
xor edx,edx
div ecx
test edx,edx
jnz .skip
jmp .no;если все числа поделились без остатка - значит числа не взаимно
простые

.skip:

dec ecx;уменьшаем число на которое проверяем делимость
cmp ecx,1;если оно после уменьшения еще больше единицы, начинаем цикл
заново
ja .loop
jmp .end;если цикл заканчивается сам то числа взаимно простые и нам ничего
не надо делать, только выйти из функции
.no;если они не взаимно простые то устанавливаем ответ как не простые
mov [prime],0
.end:
ret;выход из функции

section '.idata' import data readable

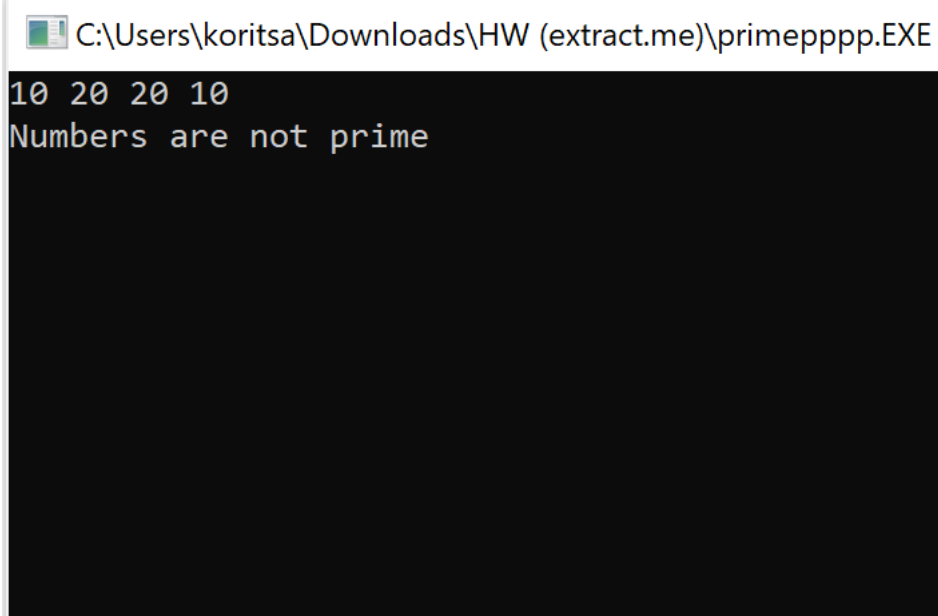
```
library kernel, 'kernel32.dll',\  
msvcrt, 'msvcrt.dll'
```

```
import kernel,\  
ExitProcess, 'ExitProcess'
```

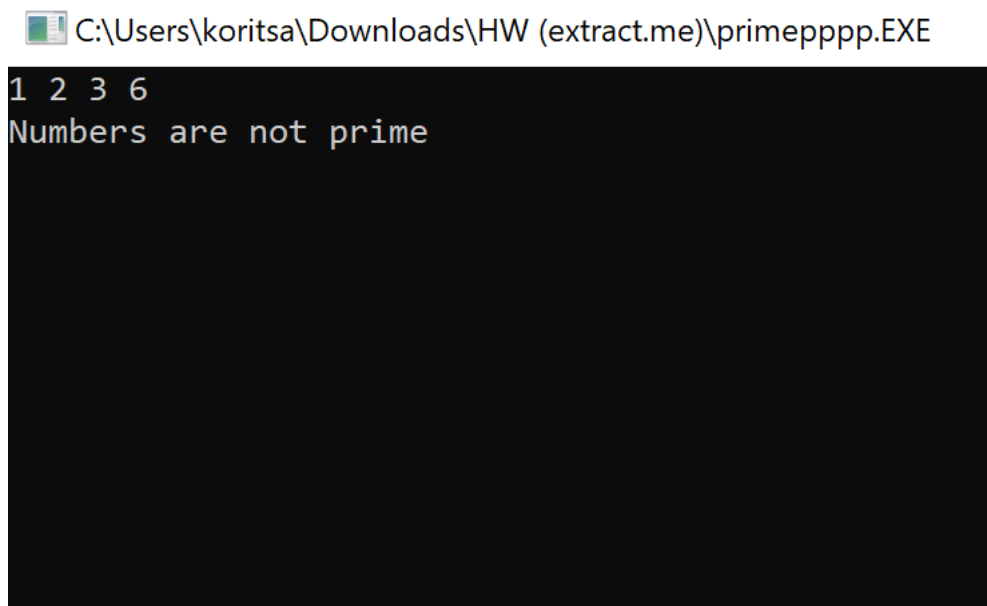
```
import msvcrt,\  
printf, 'printf',\  
scanf, 'scanf',\  
getch, '_getch'
```

ТЕСТИРОВАНИЕ ПРОГРАММЫ


1. Простая проверка работоспособности программы




```
C:\Users\koritsa\Downloads\HW (extract.me)\primepppp.EXE  
10 20 20 10  
Numbers are not prime
```




```
C:\Users\koritsa\Downloads\HW (extract.me)\primepppp.EXE  
1 2 3 6  
Numbers are not prime
```

 C:\Users\koritsa\Downloads\HW (extract.me)\primepppp.EXE

```
5 10 17 51  
Numbers are prime
```


 C:\Users\koritsa\Downloads\HW (extract.me)\primepppp.EXE

```
2 3 5 7  
Numbers are prime
```

 C:\Users\koritsa\Downloads\HW (extract.me)\primepppp.EXE

```
13 17 19 31  
Numbers are prime
```

2. Некорректный ввод

 C:\Users\koritsa\Downloads\HW (extract.me)\prime.EXE

```
0 0 0 0 0 0
```

```
Incorrect input_
```