

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук Департамент программной инженерии
Дисциплина: «Архитектура вычислительных систем»

РЕШЕТО ЭРАТОСФЕНА МИКРОПРОЕКТ

Пояснительная записка

Выполнил:
Фомин Иван,
студент гр. БПИ197.

Москва
2020

Содержание

1. Текст задания	2
2. Применяемые расчетные методы	2
2.1. Теория решения задания.....	2
3. Тестирование программы	3
ПРИЛОЖЕНИЕ 1. Список литературы	4
ПРИЛОЖЕНИЕ 2. Код программы.....	5

1. Текст задания

Разработать программу, вычисляющую простые числа в диапазоне до беззнакового двойного машинного слова по методу "Решето Эратосфена Киренского"

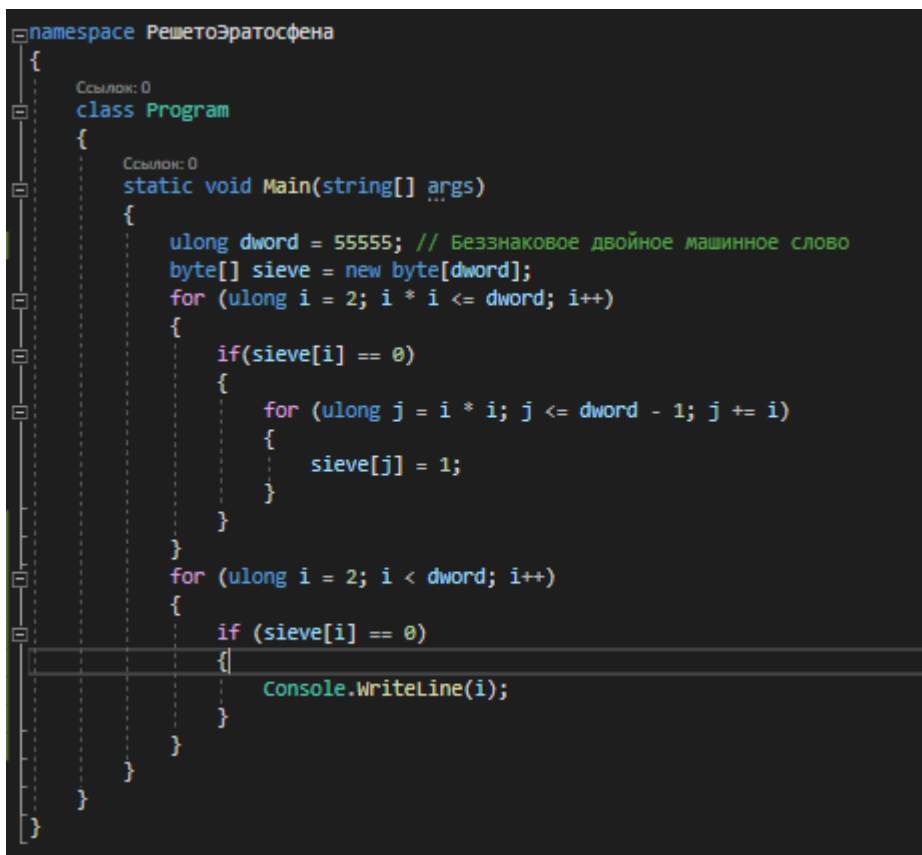
2. Применяемые расчетные методы

2.1. Теория решения задания

Для нахождения всех простых чисел не больше заданного числа n , следуя методу Эратосфена, нужно выполнить следующие шаги:

1. Выписать подряд все целые числа от двух до n (2, 3, 4, ..., n). (Массив заполненный нулями. Числа это индексы)
2. Пусть переменная p изначально равна двум — первому простому числу.
3. Зачеркнуть в списке числа от $2p$ до n считая шагами по p (это будут числа кратные p : $2p$, $3p$, $4p$, ...). (В массиве меняем 0 на 1)
4. Найти первое незачёркнутое число в списке, большее чем p , и присвоить значению переменной p это число. (Находим первое число равное нулю)
5. Повторять шаги 3 и 4, пока возможно.

Теперь все незачёркнутые числа в списке — это все простые числа от 2 до n . (Все числа равные 1му являются простыми. Выводим их)



```

namespace РешетоЭратосфена
{
    Ссылка: 0
    class Program
    {
        Ссылка: 0
        static void Main(string[] args)
        {
            ulong dword = 55555; // Беззнаковое двойное машинное слово
            byte[] sieve = new byte[dword];
            for (ulong i = 2; i * i <= dword; i++)
            {
                if(sieve[i] == 0)
                {
                    for (ulong j = i * i; j <= dword - 1; j += i)
                    {
                        sieve[j] = 1;
                    }
                }
            }
            for (ulong i = 2; i < dword; i++)
            {
                if (sieve[i] == 0)
                {
                    Console.WriteLine(i);
                }
            }
        }
    }
}

```

Рисунок 1. Пример программы на C#

3. Тестирование программы

На вход программы не подается никаких значений. Программа выводит список простых чисел в диапазоне от 2 до 2^{32}



Рисунок 2. Результат работы программы

ПРИЛОЖЕНИЕ 1

Список литературы

1. Команды безусловного и условного переходов на языке Ассемблер//
<https://sites.google.com/site/sistprogr/lekcii1/lek9> (31.10.2020)
2. Решето Эратосфена [Википедия] //
https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D0%B5%D1%82%D0%BE_%D0%AD%D1%80%D0%B0%D1%82%D0%BE%D1%81%D1%84%D0%B5%D0%BD%D0%B0 (31.10.2020)

ПРИЛОЖЕНИЕ 2

Код программы

```
format PE console
```

```
entry start
```

```
include 'win32a.inc'
```

```
section '.data' data readable writable
```

```
indexOut db '%d',10,0 ; Форма вывода в консоль
```

```
vec_size dd 50000 ; Размер массива. Находит все простые числа до  
этого числа
```

```
i dd ? ; Индексатор для заполнения массива нулями
```

```
j dd 1 ; Индексатор для первого for
```

```
k dd 1 ; Индексатор для второго for
```

```
l dd ? ; Индексатор для 3его for, который выводит простые  
числа в консоль
```

```
vec rd 500000
```

```
section '.code' code readable executable
```

```
start:
```

```
getVecLoop: ;Заполняю массив нулями
```

```
cmp ecx, [vec_size]
```

```
jge endInputVector ; Выход из цикла
```

```
mov [i], ecx
```

```
mov eax, [i]
```

```

mov dword[ebx + eax*4], 0    ;Присваиваю элементу массива 0
mov ecx, [i]
inc ecx    ;Инкрементация
jmp getVecLoop
endInputVector:

```

```

for1:    ; Находит следующей ближайшей элемент массива равный 0
add [j], 1
mov ebx, [j]
mov eax, [j]
mul eax ; умножение eax на вписанный аргумент
mov [k], eax
mov ecx, [j]
mov eax, vec
mov eax, [eax + ecx*4]
cmp eax, 0
je for2
comeBack1:

```

;----- Условия завершения цикла

```

mov eax, [j]
mul eax
cmp eax, [vec_size]
jle for1

```

```

for3:    ; Еще раз прогоняет массив и вызывает метод вывода простых
чисел в консоль
add [l], 1

```

```

mov eax, [1]
mov ebx, vec
mov ebx, dword[ebx + eax*4]
cmp ebx, 0
je output
comeBack2:
;----- Условия завершения цикла
mov eax, [1]
cmp eax, [vec_size]
jle for3
call finish

for2:      ; Заменяет все элементы массива кратные данному на 1
mov ecx, [k]
mov eax, vec
mov dword[eax + ecx*4], 1
mov eax, [k]
mov ecx, [vec_size]
sub ecx, 1
add [k], ebx
cmp eax, ecx
jle for2
jmp comeBack1

; Метод выводит простые числа в консоль
output:
mov eax, [1]
push eax
push indexOut

```



```

    call[printf]
    jmp comeBack2
; Корректное завершение программы
finish:

```

```

    call [getch]

    push 0
    call [ExitProcess]

```

;----- Подключение библиотек -----;

```

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',\
        HeapCreate, 'HeapCreate',\
        HeapAlloc, 'HeapAlloc'
include 'api\kernel32.inc'
    import msvcrt,\
        printf, 'printf',\
        scanf, 'scanf',\
        getch, '_getch'

```