Высшая школа экономики

Факультет компьютерных наук

Пояснительная записка

программа, вычисляющая простые числа в диапазоне до беззнакового машинного слова по методу "Решето Эратосфена Киренского"

Выполнил: Шакуров Семен Сергеевич, БПИ194 / 2 подгруппа.

Текст задания

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

Применяемые расчетные методы

Для решения данной задачи применялся метод "Решето Эратосфена Киренского"

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

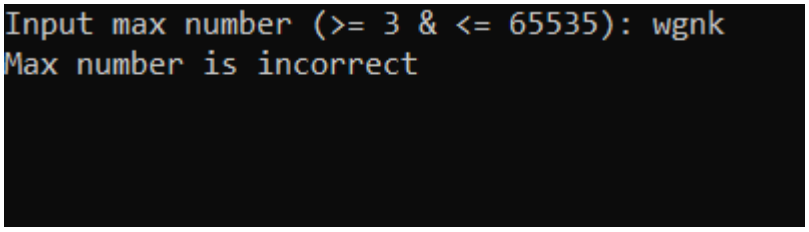
1. Выписать подряд все целые числа от двух до *n* (2, 3, 4, …, *n*).
2. Пусть переменная *p* изначально равна двум — первому простому числу.
3. Зачеркнуть в списке числа от 2*p* до *n* считая шагами по *p* (это будут числа кратные *p*: 2*p*, 3*p*, 4*p*, …).
4. Найти первое незачёркнутое число в списке, большее чем *p*, и присвоить значению переменной *p* это число.
5. Повторять шаги 3 и 4, пока возможно.

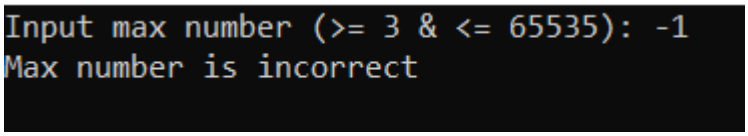
Теперь все незачёркнутые числа в списке — это все простые числа от 2 до *n*.

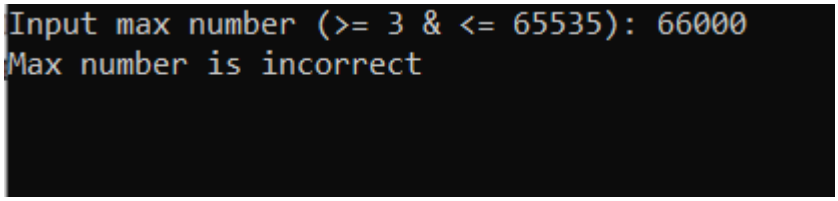
Область допустимых значений

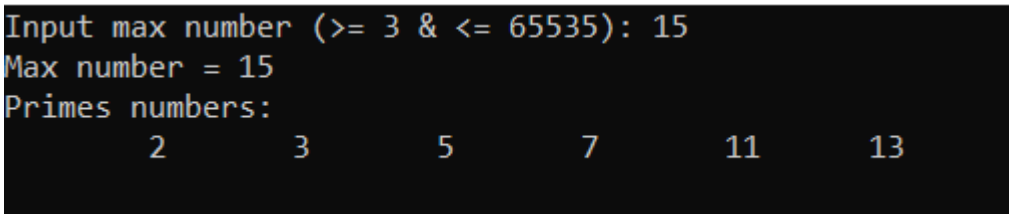
Входное значение максимального числа должно быть в диапазоне от 3 до 65535

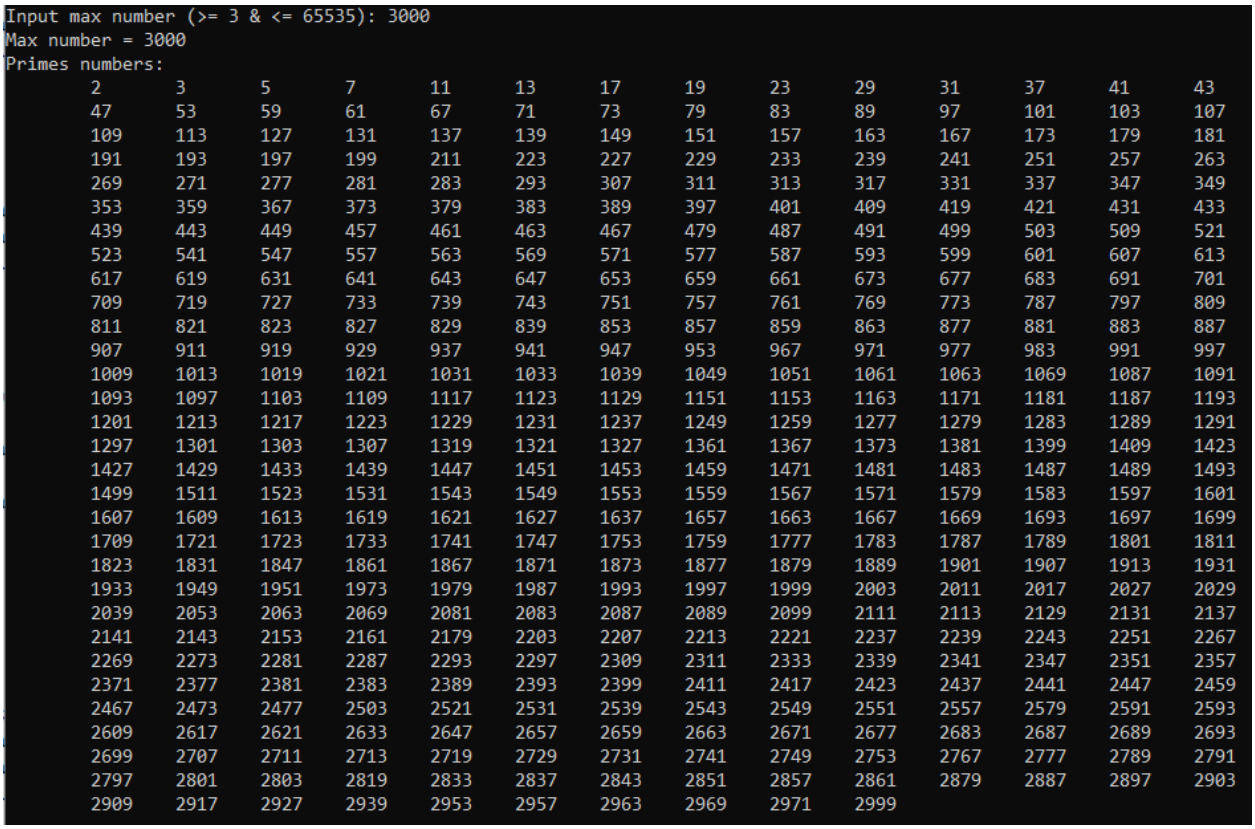
Тесты











Список используемых источников

https://ru.wikipedia.org/wiki/Решето_Эратосфена

Приложение

Текст программы

```
format PE console

entry start

include 'win32a.inc'

section '.data' data readable writeable

    maxNumberLabel db 'Input max number (>= 3 & <= 65535): ', 0
    inputFormat db '%u', 0
    maxNumError db 'Max number is incorrect', 0xD, 0xA, 0
    maxNumberOutputFormat db 'Max number = %u', 0xD, 0xA, 0
    mallocError db 'Can not allocate memory', 0xD, 0xA, 0
    primesLabel db 'Primes numbers:', 0xD, 0xA, 0
    primeOutput db '%u', 0x9, 0
    startPrint db 0x9, 0
    crlf db 0xD, 0xA, 0

    maxNumber dd 0
    primesPointer dd 0

    NULL = 0

section '.code' code readable executable

    start:
        enter 0, 0
```

```
        ;ввод максимального числа
        call inputMaxNumber
        cmp edx, 0
        jne customExit
        mov [maxNumber], eax

        ;выделяем память для массива флагов
        mov eax, [maxNumber]
        call allocateFlagsMemory
        cmp edx, 0
        jne customExit
        mov [primesPointer], eax

        ;отсеять составные числа
        mov eax, [primesPointer]
        mov ebx, [maxNumber]
        call findPrimes

        ;вывести простые числа
        mov eax, [primesPointer]
        mov ebx, [maxNumber]
        call output

        ;освободить память от массива флагов
        mov eax, [primesPointer]
        call freeFlagsMemory

        ;завершение программы
        call Exit
;-----------------------------------------------
```

```
output:
    enter 12, 1
    mov [ebp-4], eax
    mov [ebp-8], ebx

    push primesLabel
    call [printf]
    add esp, 4

    push startPrint
    call[printf]
    add esp, 4

    cld
    mov esi, [ebp-4]
    mov edx, esi
    add edx, [ebp-8]
    inc edx

    mov [ebp-12], edx
    mov ecx, 0
;Цикл для вывода
outputCicle:
    lodsb
    cmp al, 0
    jne print
    jmp isFinish

print:
    cmp ecx, 1
```

```
            je isFinish
            push esi
            push ecx
            push primeOutput
            call [printf]
            add esp, 4

            pop ecx
            pop esi
            mov edx, [ebp-12]

        isFinish:
            inc ecx
            cmp esi, edx
            jb outputCicle

            push crIf
            call [printf]
            add esp, 4

            leave
            ret
;-------------------------------------------------
        ;Поиск простых чисел с помощью решета Эратосфена
        findPrimes:
            enter 8, 1

            mov [ebp-4], eax
            add eax, ebx
            inc eax
```

```
        mov [ebp-8], eax

        ;вычеркиваем составные числа
        cld
        ;p=2
        mov edx, 2
        ;множитель c = 2
        mov ecx, 2
Cicle:
        ;x = c*p
        mov eax, edx
        push edx
        mul ecx
        pop edx

        cmp eax, ebx
        jbe strikeNumber
        jmp increase

strikeNumber:
        mov edi, [ebp-4]
        add edi, eax
        mov byte [edi], 0
        ;c++
        inc ecx
        jmp Cicle

increase:
        mov esi, [ebp-4]
        add esi, edx
```

```
        inc esi

        mov ecx, edx
        inc ecx

    checkNum:
        mov eax, ecx
        mul eax
        cmp eax, ebx
        ja returnPrimes

        lodsb
        inc ecx
        cmp al, 0
        jne newFound
        jmp checkNum

    newFound:
        mov edx, ecx
        dec edx
        mov ecx, 2
        jmp Cicle

    returnPrimes:
        leave
        ret

;-------------------------------------------------
    ;Освобидить память от массива флагов
    freeFlagsMemory:
```

```
        enter 0, 1

        push eax
        call [free]
        add esp, 4

        leave
        ret
;-------------------------------------------------
    allocateFlagsMemory:
        enter 8, 1
        ;выделить eax+1 байт
        inc eax
        mov [ebp-4], eax

        push eax
        call [malloc]
        add esp, 4

        ;проверка
        cmp eax, 0
        je fail
        mov [ebp-8], eax

        ;инициализация
        mov byte [eax], 0

        cld
        mov edi, eax
        inc edi
```

```
        mov edx, [ebp-4]
        add edx, eax

        mov al, 1

    writeTrue:
        stosb
        cmp edi, edx
        jb writeTrue

        mov eax, [ebp-8]
        jmp success

    fail:
        mov edx, mallocError
        jmp return

    success:
        mov edx, 0

    return:
        leave
        ret

;-------------------------------------------------
    ;Ввод максимального числа
    inputMaxNumber:

        enter 4, 1
```

```
        push maxNumberLabel
        call [printf]
        add esp, 4

        mov eax, ebp
        sub eax, 4

        push eax
        push inputFormat
        call [scanf]
        add esp, 8

        mov eax, [ebp - 4]

        ;проверка
        cmp eax, 3
        jb numberError
        cmp eax, 65535
        ja numberError
        jmp successInput

;выход
numberError:
        mov edx, maxNumError
        jmp returnInput

successInput:
        push eax
        push maxNumberOutputFormat
        call [printf]
```

```
        add esp, 4
        pop eax
        mov edx, 0

    returnInput:
        leave
        ret
;--------------------------------------------------------
    ;выход при ошибке
    customExit:
        push edx
        call [printf]

    ;выход
    Exit:
        call [getch]
        push NULL
        call [ExitProcess]

;--------------------------------------------------------
section '.idata' import data readable

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

    import kernel,\
        ExitProcess, 'ExitProcess'

    import msvcrt,\
        printf, 'printf',\
```

```
getch, '_getch',\
scanf, 'scanf',\
malloc, 'malloc',\
free, 'free'
```