

Федеральное государственное автономное образовательное учреждение  
высшего образования «Национальный Исследовательский Университет  
«Высшая Школа Экономики»»  
Факультет компьютерных наук  
Департамент программной инженерии

Микропроект.  
Пояснительная записка.

Выполнила  
студент БПИ196-2  
Филиппова Мария

Москва

2020

## СОДЕРЖАНИЕ

1. Постановка задачи .....	2
2. Применяемые расчетные методы .....	2
3. Входные данные .....	3
4. Испытание программы .....	3
ПРИЛОЖЕНИЕ 1 .....	11
ПРИЛОЖЕНИЕ 2 .....	27
ПРИЛОЖЕНИЕ 3 .....	29

## 1. Постановка задачи

Разработать программу реализации 4-х действий арифметики комплексных чисел, представленных дробями (использовать целые со знаком, обеспечить максимальную простоту результата).

## 2. Применяемые расчетные методы

Комплексное число – число вида  $\frac{a}{b} + \frac{c}{d} * i$ , где  $\frac{a}{b}$  – действительная часть,  $\frac{c}{d}$  – мнимая часть.

Сложение комплексных чисел:

$$\left(\frac{a_1}{b_1} + \frac{c_1}{d_1} * i\right) + \left(\frac{a_2}{b_2} + \frac{c_2}{d_2} * i\right) = \frac{a_1 * b_2 + a_2 * b_1}{b_1 * b_2} + \frac{c_1 * d_2 + c_2 * d_1}{b_2 * d_2} * i$$

Вычитание комплексного числа:

$$\left(\frac{a_1}{b_1} + \frac{c_1}{d_1} * i\right) - \left(\frac{a_2}{b_2} + \frac{c_2}{d_2} * i\right) = \frac{a_1 * b_2 - a_2 * b_1}{b_1 * b_2} + \frac{c_1 * d_2 - c_2 * d_1}{b_2 * d_2} * i$$

Вычитание можно привести к сложению, этот способ использован в программе:

$$\left(\frac{a_1}{b_1} + \frac{c_1}{d_1} * i\right) - \left(\frac{a_2}{b_2} + \frac{c_2}{d_2} * i\right) = \left(\frac{a_1}{b_1} + \frac{c_1}{d_1} * i\right) + \left(-\frac{a_2}{b_2} - \frac{c_2}{d_2} * i\right)$$

Умножение комплексных чисел:

$$\begin{aligned} \left(\frac{a_1}{b_1} + \frac{c_1}{d_1} * i\right) \times \left(\frac{a_2}{b_2} + \frac{c_2}{d_2} * i\right) \\ = \left(\frac{a_1 * a_2}{b_1 * b_2} - \frac{c_1 * c_2}{d_1 * d_2}\right) + \left(\frac{a_1 * c_2}{b_1 * d_2} + \frac{a_2 * c_1}{b_2 * d_1} * i\right) \end{aligned}$$

Умножение можно свести к сложению 2-х комплексных чисел, что и используется в программе:

$$\begin{aligned} \left(\frac{a_1 * a_2}{b_1 * b_2} + \frac{a_1 * c_2}{b_1 * d_2} * i\right) + \left(\frac{-c_1 * c_2}{d_1 * d_2} + \frac{a_2 * c_1}{b_2 * d_1} * i\right) \\ = \left(\frac{a_1 * a_2}{b_1 * b_2} - \frac{c_1 * c_2}{d_1 * d_2}\right) + \left(\frac{a_1 * c_2}{b_1 * d_2} + \frac{a_2 * c_1}{b_2 * d_1} * i\right) \end{aligned}$$

Деление комплексных чисел:

$$\frac{\frac{a_1}{b_1} + \frac{c_1}{d_1} * i}{\frac{a_2}{b_2} + \frac{c_2}{d_2} * i} = \frac{\left(\frac{a_1 * a_2}{b_1 * b_2} - \frac{c_1 * c_2}{d_1 * d_2}\right) + \left(\frac{a_1 * c_2}{b_1 * d_2} + \frac{a_2 * c_1}{b_2 * d_1} * i\right)}{\frac{a_2^2}{b_2^2} + \frac{c_2^2}{d_2^2}}$$

Операции в числителе можно свести к сложению, а затем поделить дроби действительной и мнимой частей на значение знаменателя.

Для обеспечения максимальной простоты результата числители и знаменатели дробей сокращаются на наибольший общий делитель этих чисел. Для нахождения наибольшего общего делителя был использован алгоритм Евклида:

Дано 2 натуральных числа  $a, b$ .

ПОКА  $a \neq b$

ЕСЛИ  $a > b$

$a = a - b$

ИНАЧЕ

$b = b - a$

КОНЕЦ

По окончанию алгоритма получится,  $a = b =$  значение наибольшего общего делителя.

### 3. Входные данные

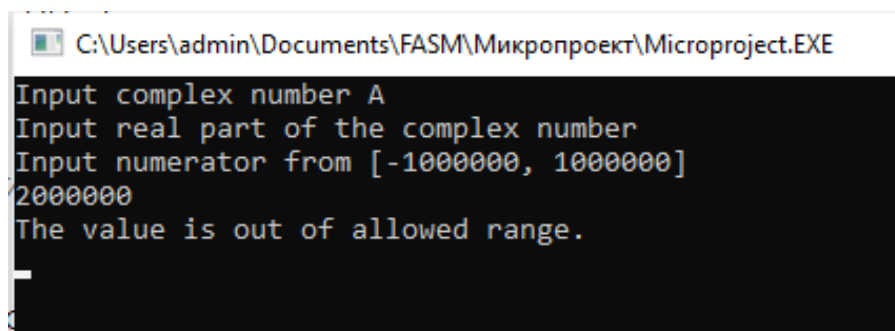
При запросе ввода числителя или знаменателя чисел – целые числа со знаком, лежащие в диапазоне  $[-1000000; 1000000]$ . При запросе ввод операции – один из символов ‘+’ – для сложения, ‘-’ – для вычитания, ‘\*’ – для умножения, ‘/’ – для деления.

### 4. Испытание программы

4.1 Область допустимых входных параметров - целочисленные значения из диапазона  $[-1000000; 1000000]$ . Пользователь будет уведомлен о попытке введения числа не из допустимых значений.

Входные данные: 2000000

Результат:



```
C:\Users\admin\Documents\FASM\Микропроект\Microproject.EXE
Input complex number A
Input real part of the complex number
Input numerator from [-1000000, 1000000]
2000000
The value is out of allowed range.
```

#### 4.2 Попытка ввести дробь с 0 в знаменателе.

Входные данные:

1

0

Результат:

```
C:\Users\admin\Documents\FASM\Микропроект\Microproject.EXE
Input complex number A
Input real part of the complex number
Input numerator from [-1000000, 1000000]
1
Input denominator from [-1000000, 1000000]
0
Attempt to divide by zero.
```

#### 4.3 Попытка ввести операцию, отличную от '+', '-', '/', '\*'.

Входные данные:

1

2

3

4

1

1

2

3

?

Результат:

```
C:\Users\admin\Documents\FASM\Микропроект\Microproject.EXE
Input complex number A
Input real part of the complex number
Input numerator from [-1000000, 1000000]
1
Input denominator from [-1000000, 1000000]
2
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
3
Input denominator from [-1000000, 1000000]
4
Input complex number B
Input real part of the complex number
Input numerator from [-1000000, 1000000]
1
Input denominator from [-1000000, 1000000]
1
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
2
Input denominator from [-1000000, 1000000]
3
Input operation
Undefined operation.
```

#### 4.4 Сложение комплексных чисел.

Входные данные:

1  
2  
3  
19  
4  
7  
3  
2  
+

Результат:

```
C:\Users\admin\Documents\FASM\Микропроект\Microproject.EXE
Input complex number A
Input real part of the complex number
Input numerator from [-1000000, 1000000]
1
Input denominator from [-1000000, 1000000]
2
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
3
Input denominator from [-1000000, 1000000]
19
Input complex number B
Input real part of the complex number
Input numerator from [-1000000, 1000000]
4
Input denominator from [-1000000, 1000000]
7
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
3
Input denominator from [-1000000, 1000000]
2
Input operation
Result:
(1/2 + 3/19 * i) + (4/7 + 3/2 * i) = (15/14 + 63/38 * i)
```

#### 4.5 Сложение комплексных чисел с 0 в результате

Входные данные:

1  
4  
-3  
5  
1  
-4  
3  
5

+

Результат:

```
C:\Users\admin\Documents\FASM\Микропроект\Microproject.EXE
Input complex number A
Input real part of the complex number
Input numerator from [-1000000, 1000000]
1
Input denominator from [-1000000, 1000000]
4
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
-3
Input denominator from [-1000000, 1000000]
5
Input complex number B
Input real part of the complex number
Input numerator from [-1000000, 1000000]
1
Input denominator from [-1000000, 1000000]
-4
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
3
Input denominator from [-1000000, 1000000]
5
Input operation
Result:
(1/4 - 3/5 * i) + (-1/4 + 3/5 * i) = (0/1 + 0/1 * i) _
```

#### 4.6 Вычитание комплексных чисел

Входные данные:

- 1
- 2
- 3
- 19
- 4
- 7
- 3
- 2
- 

Результат:

```
C:\Users\admin\Documents\FASM\Микропроект\Microproject.EXE
Input complex number A
Input real part of the complex number
Input numerator from [-1000000, 1000000]
1
Input denominator from [-1000000, 1000000]
2
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
3
Input denominator from [-1000000, 1000000]
19
Input complex number B
Input real part of the complex number
Input numerator from [-1000000, 1000000]
4
Input denominator from [-1000000, 1000000]
7
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
3
Input denominator from [-1000000, 1000000]
2
Input operation
Result:
(1/2 + 3/19 * i) - (4/7 + 3/2 * i) = (-1/14 - 51/38 * i)_
```

#### 4.7 Умножение комплексных чисел

Входные данные:

15

16

3

19

-7

8

0

1

\*

Результат:



```
C:\Users\admin\Documents\FASM\Микропроект\Microproject.EXE
Input complex number A
Input real part of the complex number
Input numerator from [-1000000, 1000000]
15
Input denominator from [-1000000, 1000000]
16
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
3
Input denominator from [-1000000, 1000000]
19
Input complex number B
Input real part of the complex number
Input numerator from [-1000000, 1000000]
-7
Input denominator from [-1000000, 1000000]
8
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
0
Input denominator from [-1000000, 1000000]
1
Input operation
Result:
(15/16 + 3/19 * i) * (-7/8 + 0/1 * i) = (-105/128 - 21/152 * i)_
```

#### 4.8 Деление комплексных чисел

Входные данные:

2

17

-3

1

2

9

5

30

/

Результат:

Выбрать C:\Users\admin\Documents\FASM\Микропроект\Microproject.EXE

```
Input complex number A
Input real part of the complex number
Input numerator from [-1000000, 1000000]
2
Input denominator from [-1000000, 1000000]
17
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
-3
Input denominator from [-1000000, 1000000]
1
Input complex number B
Input real part of the complex number
Input numerator from [-1000000, 1000000]
2
Input denominator from [-1000000, 1000000]
9
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
5
Input denominator from [-1000000, 1000000]
30
Input operation
Result:
(2/17 - 3/1 * i) / (2/9 + 5/30 * i) = (-522/85 - 756/85 * i)
```

#### 4.9 Деление комплексных чисел с попыткой деления на 0.

Входные данные:

4

18

3

9

0

-2

0

6

/

Результат:

```
C:\Users\admin\Documents\FASM\Микронпроект\Microproject.EXE
Input complex number A
Input real part of the complex number
Input numerator from [-1000000, 1000000]
4
Input denominator from [-1000000, 1000000]
18
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
3
Input denominator from [-1000000, 1000000]
9
Input complex number B
Input real part of the complex number
Input numerator from [-1000000, 1000000]
0
Input denominator from [-1000000, 1000000]
-2
Input imaginary part of the complex number
Input numerator from [-1000000, 1000000]
0
Input denominator from [-1000000, 1000000]
6
Input operation
Attempt to divide by zero.
```

## Код программы

### Microproject.ASM

```
; Выполнила: Филиппова Мария, БПИ-196.
; Задание.
; Разработать программу реализации 4-х действий
; арифметики комплексных чисел, представленных дробями
; (использовать целые со знаком, обеспечить максимальную
; простоту результата).
format PE console
entry start

include 'win32a.inc'
include 'Microproject.inc'

;-----
-----
section '.data' data readable writable
    ; подсказки пользователю при вводе данных
    strInputA db 'Input complex number A', 10, 0
    strInputB db 'Input complex number B', 10, 0
    strInputRe db 'Input real part of the complex
number', 10, 0
    strInputIm db 'Input imaginary part of the complex
number', 10, 0
    strInputNumerator db 'Input numerator from [-
1000000, 1000000]', 10, 0
    strInputDenominator db 'Input denominator from [-
1000000, 1000000]', 10, 0
    strIncorrectVal db 'Attempt to divide by zero.', 10,
0
    strIncorrectOp db 'Undefined operation.', 10, 0
    strOutOfRange db 'The value is out of allowed
range.', 10, 0
    strScanInt db '%d', 0
    strEmpty db ' ', 0
    strOp db 'Input operation', 10, 0

    d dd ? ; НОД
```

```

; Зрание числителей, знаменателей дробей,
; входящих в состав комплексных чисел.
; A:
a_re_num dd ?
a_re_den dd ?
a_im_num dd ?
a_im_den dd ?
; B :
b_re_num dd ?
b_re_den dd ?
b_im_num dd ?
b_im_den dd ?
; C, здесь же хранится результат операций:
c_re_num dd ?
c_re_den dd ?
c_im_num dd ?
c_im_den dd ?
;
; Резервирование места для хранения кода операции
op rd 1
; Строки оформления результата.
strRes db 'Result: ', 10, 0
strOpAdd db ' + ', 0
strOpSub db ' - ', 0
strOpMul db ' * ', 0
strOpDiv db ' / ', 0
strOpEq db ' = ', 0
strCNPosIm db '(%d/%d + %d/%d * i)', 0
strCNNegIm db '(%d/%d - %d/%d * i)', 0

; Промежуточные значения при операциях
умножения/деления.
x_re_num dd ?
x_re_den dd ?
x_im_num dd ?
x_im_den dd ?
y_re_num dd ?
y_re_den dd ?
y_im_num dd ?
y_im_den dd ?

```

```

;-----
-----
section '.code' code readable executable

start:
    ;Ввод комплексных чисел А и В.
    CNInputMacro strInputA, a_re_num, a_re_den, a_im_num,
a_im_den
    CNInputMacro strInputB, b_re_num, b_re_den, b_im_num,
b_im_den

    ;Ввод действия.
    Print strOp

    call [getch]
    mov [op], eax
    ; 43 = '+', 45 = '-', 42 = '*', 47 = '/'
    ; Выполнить действие в зависимости от введенного действия.
    cmp [op], 43 ; +
    jne @notAdd
        CNAddMacro
        jmp printResult
@notAdd:
    cmp [op], 45 ; -
    jne @notSub
        CNSubMacro
        jmp printResult
@notSub:
    cmp [op], 42 ; *
    jne @notMul
        CNMulMacro
        jmp printResult
@notMul:
    cmp [op], 47 ; /
    jne @notDiv
        CNDivMacro
        jmp printResult
@notDiv:

```

```

        ; В случае некорректно введенного оператора, выводится
        ; сообщение об ошибке.
        Print strIncorrectOp
        jmp finish

; Вывод результата.
printResult:

        ; проверить, не появился ли 0 в дробях знаменателя
        ; результата
        cmp [c_re_den], 0
        je DivByZero

        cmp [c_im_den], 0
        je DivByZero

        ; Приводим дроби к формату, где знаменатель
        ; положителен, а
        ; знак числителя отображает знак всей дроби.
        formatFractions a_re_num, a_re_den, a_im_num, a_im_den
        formatFractions b_re_num, b_re_den, b_im_num, b_im_den
        formatFractions c_re_num, c_re_den, c_im_num, c_im_den

        ; Вывод результата.
        Print strRes
        OutputResult
        jmp finish

; Уведомление о попытке деления на 0.
DivByZero:
        Print strIncorrectVal

finish:
        call [getch]
        push 0
        call [ExitProcess]

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

```

```

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

import kernel,\
             ExitProcess, 'ExitProcess'

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

```

### **Microproject.inc**

```

; Вывод значений.
macro Print [args] {
    reverse
    push args
    common
    call [printf]
}

; Ввод значений.
macro Scan [args] {
    reverse
    push args
    common
    call [scanf]
}

; Ввод дроби.
macro GetFraction num, den {
    local getFraction, outOfRange, endGetFraction
    getFraction:
        ; Ввод числителя.
        Print strInputNumerator
        Scan strScanInt, num

        ; Проверка чилителя на корректность.
        cmp [num], 1000000
        jg outOfRange

        cmp [num], -1000000

```



```

    j1 outOfRange

; Ввод знаменателя.
Print strInputDenominator
Scan strScanInt, den

; Проверка знаменателя на корректность.
cmp [den], 1000000
jg outOfRange

cmp [den], -1000000
j1 outOfRange

; Проверка на неравенство знаменателя 0.
cmp [den], 0
jne endGetFraction

Print strIncorrectVal
call [getch]
push 0
call [ExitProcess]

outOfRange:
    Print strOutOfRange
    call [getch]
    push 0
    call [ExitProcess]

endGetFraction:

}

; Ввод одного комплексного числа.
macro CNInputMacro strInput, n_re_num, n_re_den, n_im_num,
n_im_den {
    Print strInput
    Print strInputRe
    ; Ввод действительной части.
    GetFraction n_re_num, n_re_den

```

```

    Print strInputIm
    ; Ввод мнимой части.
    GetFraction n_im_num, n_im_den
}

;Сокращение дроби.
macro ReduceFraction first, second{
local startNOD,negB, negA, negBB, @greaterB, NODLoop,
negDivA, negDivB, negDivBB, reduceFraction, redIfZeroNum,
endNOD
startNOD:
    ; НОД - это целое натуральное число,
    ; поэтому проверяем числа на отрицательность и на
    равенство 0,
    ; чтобы искать НОД для 2-х положительных чисел.
    cmp [first],0
    je redIfZeroNum

    mov eax, [first]
    mov ebx, [second]
    cmp [first], 0
    jl negA
    cmp [second], 0
    jl negB
    jmp NODLoop
negB:
    neg ebx
    jmp NODLoop
negA:
    neg eax
    cmp [second], 0
    jl negBB
    jmp NODLoop
negBB:
    neg ebx
    jmp NODLoop

; Алгоритм Евклида.
NODLoop:
    cmp eax, ebx

```

```
je reduceFraction
```

```
cmp eax, ebx
jl @greaterB
    sub eax, ebx
    jmp NODLoop
@greaterB:
    sub ebx, eax
```

```
jmp NODLoop
```

; Сокращение числителя и знаменателя на их НОД.

reduceFraction:

```
mov [d], eax
Print strEmpty
```

```
;поделить на d оба числа
;положительное число делится адекватно,
;отрицательное нужно сделать положительным, а затем
;поменять знакрезультата.
```

```
cmp [first], 0
jl negDivA
    mov eax, [first]
    div [d]
    mov [first], eax
```

```
cmp [second], 0
jl negDivB
    mov eax, [second]
    div [d]
    mov [second], eax
    jmp endNOD
```

```
negDivB:
    mov eax, [second]
    neg eax
    div [d]
    neg eax
    mov [second], eax
    jmp endNOD
```

```
negDivA:
```

```

mov eax, [first]
neg eax
div [d]
neg eax
mov [first], eax

cmp [second], 0
j1 negDivBB
    mov eax, [second]
    div [d]
    mov [second], eax
    jmp endNOD
negDivBB:
    mov eax, [second]
    neg eax
    div [d]
    neg eax
    mov [second], eax
    jmp endNOD

```

; В случае равенства числителя дроби 0,  
; присвоим знаменателю значение 0.

redIfZeroNum:

```

    mov [second], 1

```

endNOD:

}

; Сложение дробей.

macro AddFractions a, b, c, d, num, den{

; Расчет числителя.

```

    mov eax, [a]
    imul eax, [d]

```

```

    mov ebx, [b]
    imul ebx, [c]

```

```

    add eax, ebx
    mov [num], eax

```

```

        ; Расчет знаменателя.
        mov eax, [b]
        imul eax, [d]
        mov [den], eax
    }

; Сложение комплексных чисел.
macro CNAAddMacro{
    ; Сложение действительных частей.
    AddFractions a_re_num, a_re_den, b_re_num, b_re_den,
    c_re_num, c_re_den

    ReduceFraction c_re_num, c_re_den

    ; Сложение мнимых частей.
    AddFractions a_im_num, a_im_den, b_im_num, b_im_den,
    c_im_num, c_im_den

    ReduceFraction c_im_num, c_im_den

}

; Отрицание числа.
macro negNum num{
    mov eax, [num]
    neg eax
    mov [num], eax
}

; Вычитание комплексных чисел.
macro CNSubMacro{
    ; - это тот же плюс, только с отрицанием b_re_num и
    b_im_num
    negNum b_re_num
    negNum b_im_num

    CNAAddMacro

    negNum b_re_num
    negNum b_im_num

```

```
}
```

```
; Присвоить secondValue значение firstValue.
```

```
macro changeValues firstValue, secondValue{
```

```
    mov eax, [firstValue]
```

```
    mov [secondValue], eax
```

```
}
```

```
;Присваиваем одному кумплексному числу значение другого.
```

```
macro modify s_re_num, s_re_den, s_im_num, s_im_den,
```

```
d_re_num, d_re_den, d_im_num, d_im_den {
```

```
    changeValues s_re_num, d_re_num
```

```
    changeValues s_re_den, d_re_den
```

```
    changeValues s_im_num, d_im_num
```

```
    changeValues s_im_den, d_im_den
```

```
}
```

```
; Умножение комплексных чисел.
```

```
масро CNMulMacro{
```

```
    ; Сохраним значения чисел a, b в x, y, чтобы потом можно  
    было их восстановить
```

```
    ; В силу реализации сложения, мы можем оперировать  
    только a и b
```

```
    ; Можно было, конечно, добавить параметры в сложение,
```

```
    ; но показалось неразумным нагромождать операции  
    сложения и вычитания.
```

```
    ; Сохраняем a в x, b в y.
```

```
    modify a_re_num, a_re_den, a_im_num, a_im_den, x_re_num,  
x_re_den, x_im_num, x_im_den
```

```
    modify b_re_num, b_re_den, b_im_num, b_im_den, y_re_num,  
y_re_den, y_im_num, y_im_den
```

```
    ; Действительная часть новых дробей.
```

```
    mov eax, [a_re_num]
```

```
    imul eax, [b_re_num]
```

```
    mov [a_re_num], eax
```

```
    mov eax, [a_re_den]
```

```
    imul eax, [b_re_den]
```

```
mov [a_re_den], eax
```

```
negNum a_im_num  
mov eax, [a_im_num]  
imul eax, [b_im_num]  
mov [b_re_num], eax  
negNum a_im_num
```

```
mov eax, [a_im_den]  
imul eax, [b_im_den]  
mov [b_re_den], eax
```

```
; Мнимая часть новых дробей.
```

```
mov eax, [x_re_num]  
imul eax, [y_im_num]  
mov [a_im_num], eax
```

```
mov eax, [x_re_den]  
imul eax, [y_im_den]  
mov [a_im_den], eax
```

```
mov eax, [x_im_num]  
imul eax, [y_re_num]  
mov [b_im_num], eax
```

```
mov eax, [x_im_den]  
imul eax, [y_re_den]  
mov [b_im_den], eax
```

```
CNAddMacro
```

```
; Возвращение первоначальных значений a и b.
```

```
modify x_re_num, x_re_den, x_im_num, x_im_den, a_re_num,  
a_re_den, a_im_num, a_im_den  
modify y_re_num, y_re_den, y_im_num, y_im_den, b_re_num,  
b_re_den, b_im_num, b_im_den  
}
```

```
; Деление комплексных чисел.
```

```

macro CNDivMacro{
    ; Числитель - умножение a и сопряженного к b.
    negNum b_im_num
    CNMulMacro
    negNum b_im_num

    ; Знаменатель сумма квадратов действительной и мнимой
    части b.
    ; Проверяем, не равен ли b 0, в таком случае, деление
    на 0 невозможно.
    cmp [b_re_num], 0
    je divByZero

    cmp [b_im_num], 0
    je divByZero

    mov eax, [b_re_num]
    imul eax, [b_re_num]
    mov [x_re_num], eax

    mov eax, [b_re_den]
    imul eax, [b_re_den]
    mov [x_re_den], eax

    mov eax, [b_im_num]
    imul eax, [b_im_num]
    mov [y_re_num], eax

    mov eax, [b_im_den]
    imul eax, [b_im_den]
    mov [y_re_den], eax

    AddFractions x_re_num, x_re_den, y_re_num, y_re_den,
    x_im_num, x_im_den

    ; Делим числитель на знаменатель
    mov eax, [c_re_num]
    imul eax, [x_im_den]
    mov [c_re_num], eax

```



```

mov eax, [c_re_den]
imul eax, [x_im_num]
mov [c_re_den], eax

mov eax, [c_im_num]
imul eax, [x_im_den]
mov [c_im_num], eax

mov eax, [c_im_den]
imul eax, [x_im_num]
mov [c_im_den], eax

; Сокращаем дроби результатов.
ReduceFraction c_re_num, c_re_den
ReduceFraction c_im_num, c_im_den
jmp endDiv

```

divByZero:

```

Print strIncorrectVal
call [getch]
push 0
call [ExitProcess]

```

endDiv:

```

}
```

; Форматирование дроби по правилу  $\text{sqn}(\text{num} * \text{den}) * |\text{num}| / |\text{den}|$

```

macro checkFraction num, den{
local startCheckFract,@denLess, endCheckFraction

```

startCheckFract:

```

    cmp [den], 0
    jl @denLess
    jmp endCheckFraction

```

@denLess:

```

    mov eax, [num]
    neg eax
    mov [num], eax

```

```

    mov eax, [den]

```

```

neg eax
mov [den], eax
jmp endCheckFraction

```

endCheckFraction:

```

}
; Форматирование дробей комплексного числа.
macro formatFractions re_num, re_den, im_num, im_den{
    ; Приводим дроби к виду  $\frac{\text{sqn}(\text{num}, \text{den}) * |\text{num}|}{|\text{den}|}$ 
    checkFraction re_num, re_den
    checkFraction im_num, im_den
}

```

```

; Вывод комплексного числа.
macro outputCN re_num, re_den, im_num, im_den{

```

```

local startOutputCN, negIm, endOutputCN

```

```

startOutputCN:
    ; Выбор строки формата, в зависимость от знака мнимой
    части.

```

```

    cmp [im_num], 0
    jnl negIm
    Print strCNPosIm, [re_num], [re_den], [im_num],
[im_den]
    jmp endOutputCN

```

```

negIm:
    mov eax, [im_num]
    neg eax
    mov [im_num], eax
    Print strCNNegIm, [re_num], [re_den], [im_num],
[im_den]
    jmp endOutputCN

```

```

endOutputCN:
}

```

```

; Вывод результата в формате A op B = C, где A, B, C -
комплексные числа,
; a op - введенный оператор.
macro OutputResult{
startOutput:
    ; Вывод первого введенного числа.
    outputCN a_re_num, a_re_den, a_im_num, a_im_den

    ; Выбор строкового представления оператора в
зависимости от кода.
    ; 43 = '+', 45 = '-', 42 = '*', 47 = '/'
    cmp [op], 43
    jne notAddOp
        Print strOpAdd
        jmp endOutput
    notAddOp:
        cmp [op], 45
        jne notSubOp
            Print strOpSub
            jmp endOutput
        notSubOp:
            cmp [op], 42
            jne notMultOp
                Print strOpMul
                jmp endOutput
            notMultOp:
                Print strOpDiv
    endOutput:
        ; Вывод второго введенного числа.
        outputCN b_re_num, b_re_den, b_im_num, b_im_den
        Print strOpEq
        ; Вывод результата.
        outputCN c_re_num, c_re_den, c_im_num, c_im_den
}

```

Переменные

Название	назначение
a_im_den	Знаменатель дроби мнимой части первого введенного комплексного числа.
a_im_num	Числитель дроби мнимой части первого введенного комплексного числа.
a_re_den	Знаменатель дроби действительной части первого введенного комплексного числа.
a_re_num	Числитель дроби действительной части первого введенного комплексного числа.
b_im_den	Знаменатель дроби мнимой части второго введенного комплексного числа.
b_im_num	Числитель дроби мнимой части второго введенного комплексного числа.
b_re_den	Знаменатель дроби действительной части второго введенного комплексного числа.
b_re_num	Числитель дроби действительной части второго введенного комплексного числа.
c_im_den	Знаменатель дроби мнимой части результата.
c_im_num	Числитель дроби мнимой части результата.
c_re_den	Знаменатель дроби действительной части результата.
c_re_num	Числитель дроби действительной части результата.
d	Наибольшее общий делитель
op	Код введенной операции
strCNNegIm	Строка вывода комплексного числа при отрицательной мнимой части.
strCNPosIm	Строка вывода комплексного числа при положительной мнимой части.
strEmpty	Пустая строка.
strIncorrectOp	Уведомление о введенной некорректной операции.
strIncorrectVal	Уведомление о попытке деления на 0.
strInputA	Уведомление о вводе первого комплексного числа.
strInputB	Уведомление о вводе второго комплексного числа.
strInputDenominator	Уведомление о вводе знаменателя дроби.
strInputIm	Уведомление о вводе мнимой части комплексного числа.
strInputNumerator	Уведомление о вводе числителя.
strInputRe	Уведомление о вводе действительной части комплексного числа.
strOp	Уведомление о вводе оператора.
strOpAdd	Строковое представление операции сложения.
strOpDiv	Строковое представление операции деления.

strOpEq	Строковое представление равенства.
strOPMul	Строковое представление операции умножения.
strOpSub	Строковое представление операции вычитания.
strOutOfRange	Уведомление о вводе числа вне допустимого диапазона.
strRes	Уведомление о выводе результата.
strScanInt	Строка для отображения вводимого числа.
x_im_den	Значения числителей и знаменателей дробей комплексных чисел, используемых для хранения первоначальных введенных чисел во время операций умножения и деления.
x_im_num	
x_re_den	
x_re_num	
y_im_den	
y_im_num	
y_re_den	
y_re_num	

Макроопределения

Название	назначение
AddFractions	Сложение дробей.
changeValues	Присваивание одному числу значение другого числа.
checkFraction	Форматирование дроби, при котором числителю присваивается знак дроби, а знаменатель становится положительным.
CNAddMacro	Сложение комплексных чисел.
CNDivMacro	Деление комплексных чисел.
CNInputMacro	Ввод одного комплексного числа.
CNMulMacro	Умножение комплексных чисел.
CNSubMacro	Вычитание комплексных чисел.
formatFractions	Форматирование дробей комплексного числа.
GetFraction	Ввод дроби, проверка на корректность значений числителя и знаменателя.
modify	Присваивание одного комплексному комплексному числу значения другого.
negNum	Перемена знака числа.
outputCN	Вывод комплексного числа.
OutputResult	Вывод строки, состоящей из действия и результата действия.
Print	Вывод переданных значений в консоль.
ReduceFraction	Сокращение дроби путем деления числителя и знаменателя на рассчитанный НОД.
Scan	Ввод значений с консоли.