

Języki Skryptowe  
dokumentacja projektu  
ZADANIE 1 – LICZBY ZESPOLONE  
konkurs Algorytmion 2017

Patryk Sroczyński, grupa 4H

16 stycznia 2022

# Część I

## Opis programu

Liczbą zespoloną nazywamy wyrażenie postaci  $a + bi$  gdzie  $a$  i  $b$  są dowolnymi liczbami rzeczywistymi, a  $i$  jest jednostką urojoną, spełniającą warunek  $i^2 = -1$  (jak widać,  $i$  nie może być liczbą rzeczywistą).

Napisz program, który po podaniu dwóch liczb zespolonych  $x$  i  $y$  oraz liczby naturalnej  $n$ , zwracał będzie sumę  $x + y$ , różnicę  $x - y$ , iloczyn  $x * y$ , iloraz  $x/y$  liczb zespolonych  $x$  i  $y$  oraz  $n$ -tą potęgę liczby zespolonej  $x$ . Przykładowo, dla danych:  $x = 3 + 2i$ ,  $y = 2 - 3i$ ,  $n = 3$ , program zwróci kolejno:  $5 - i$ ,  $1 + 5i$ ,  $12 - 5i$ ,  $i$  oraz  $9 + 46i$ .

Wskazówka (do dzielenia liczb zespolonych): sprawdź, jaką ciekawą cechę ma iloczyn dwóch liczb zespolonych  $a + bi$  oraz  $a - bi$ .

Sposób wprowadzania liczb zespolonych pozostawiamy w gestii rozwiązującego

## Instrukcja obsługi

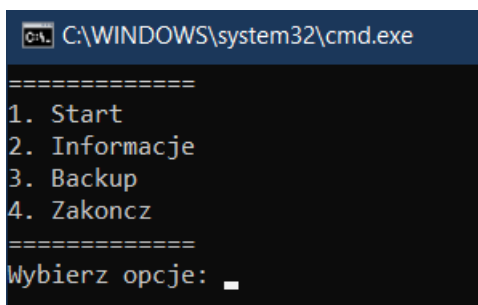
Aby wykonać program, należy uruchomić skrypt app.bat. Ponadto w folderze powinny znajdować się dwa skrypty pythonowe (app.py oraz web.py) oraz plik z danymi wejściowymi (in.txt) zawierający 5 liczb wpisanych jedna pod drugą.

Program posiada 4 opcje: Start, Informacje, Backup, Zakoncz.

Uruchomienie pierwszej opcji powoduje wywołanie się pierwszego skryptu app.py. Skrypt ten przetwarza dane z pliku in.txt, a następnie tworzy out.txt. W kolejnym kroku program wywołuje drugi skrypt web.py, który przetwarza te dane i wyświetla je na stronie.

Opcja informacje wyświetla dane o projekcie.

Backup tworzy kopie wszystkich plików z folderu, w przypadku gdy backup już istnieje to nadpisuje stare pliki.



```
C:\WINDOWS\system32\cmd.exe
=====
1. Start
2. Informacje
3. Backup
4. Zakoncz
=====
Wybierz opcje: _
```

Rysunek 1: MENU programu

Działania na liczbach zespolonych			
Dodawanie	$4.0 + 1.0i$	$2.0 + 3.0i$	$6 + 4i$
Odejmowanie	$4.0 + 1.0i$	$2.0 + 3.0i$	$2 - 2i$
Mnożenie	$4.0 + 1.0i$	$2.0 + 3.0i$	$5 + 14i$
Dzielenie	$4.0 + 1.0i$	$2.0 + 3.0i$	$1 - 1i$
Potegowanie	$4.0 + 1.0i$		$52 + 47i$
Srednia wynikow			$13.2 + 12.4i$

Rysunek 2: Wynik działań

Efekt wywołanego skryptu web.py. Skrypt tworzy stronę HTML zawierającą tabele z wynikami działań. W przypadku braku niektórych danych funkcja wpisuje w jej miejsce wartość 0.

### Informacje dodatkowe

- Skrypty pythonowe zostały zaimplementowane w wersji Python 3.7.11
- Program wyrzuca wyjątek w przypadku braku danych wejściowych
- W przypadku nie wystarczającej ilości danych wejściowych program przypisuje wartość 1
- Wynik działań przetwarzany jest na stronę HTML

## Część II

### Opis działania

Liczby zespolone są rozszerzeniem liczb rzeczywistych  $R$ . Zbiór liczb zespolonych oznaczamy symbolem  $C$ .

Liczbę urojoną zapisujemy za pomocą jednostki urojonej  $i$ . Liczbę  $i$  definiujemy następująco:

$$i^2 = -1$$

Liczbę zespoloną ogólnie możemy zapisać:

$$a + bi$$

Gdzie:  $a$  to część rzeczywista,  $b$  to część urojona, a  $i$  to jednostka urojona

Dodawanie, odejmowanie, mnożenie liczb zespolonych wykonujemy tak jak wyrażenia algebraiczne w dziedzinie  $R$ , jednocześnie pamiętając, że  $i^2 = -1$ . Natomiast przy dzieleniu skorzystamy z następującego wzoru:

$$\frac{z1}{z2} = \frac{(ac + bd)}{|z2|^2} + \frac{(bc + ad)}{|z2|^2}i$$

Podczas potęgowania skorzystamy ze wzoru de Moivre'a:  $z^n = |z|^n(\cos n\phi + i\sin n\phi)$ , gdzie  $\phi$  - argument liczby zespolonej,  $|z|$  - moduł liczby zespolonej

### Implementacja

Liczby zespolone oparte są na klasie `Complex`. Klasa ta przyjmuje w swoim konstruktorze dwa parametry. Jest to kolejno część rzeczywista liczby i część urojona. Do wykonywania działań arytmetycznych wykorzystane zostały tzw. magic methods. Są to metody które w zapisie zaczynają się oraz kończą podwójnym podkreśleniem (np. `__pow__()`). Metody te nie są wywoływane bezpośrednio przez użytkownika, natomiast zostają wykonane automatycznie w zależności od wykonywanych działań. Np. metoda `__add__()` zostanie wywołana, gdy dodamy dwie liczby zespolone.

```
1 class Complex(object):
2
3     def __init__(self, real, imag = 0.0):
4         self.real = real
5         self.imag = imag
6
7     def __add__(self, other):
8         return Complex(self.real + other.real, self.imag + other.imag)
9
10    def __sub__(self, other):
11        return Complex(self.real - other.real, self.imag - other.imag)
12
13    def __mul__(self, other):
```

```

14         return Complex((self.real * other.real) - (self.imag * other.
15             imag),
16             (self.imag * other.real) + (self.real * other.imag))
17     def __truediv__(self, other):
18         m = other.real ** 2 + other.imag ** 2
19         return Complex((self.real * other.real + self.imag * other.imag)
20             / m,
21             (self.imag * other.real - self.real * other.imag) / m)
22     def __pow__(self, n):
23         r = pow(math.hypot(self.real, self.imag), n)
24         phi = n * ((math.pi / 2 - math.atan2(self.real, self.imag)) % (
25             math.pi * 2))
26         return Complex(math.cos(phi) * r, math.sin(phi) * r)

```

---

## Pseudokod

**Data:** Dane wejściowe plik in.txt *in*

*numbers* := [];

*counter* := 0;

**while** *in* **do**

*numbers*[*counter*] = *in*;

*counter* ++;

**if** *numbers*[*counter*] == 0 **then**

*numbers*[*counter*] = 1;

**end**

**end**

*a* := *Zespolone*(*numbers*[0], *numbers*[1]);

*b* := *Zespolone*(*numbers*[2], *numbers*[3]);

*n* := *numbers*[4];

*suma* = *a* + *b*;

*roznica* = *a* - *b*;

*iloczyn* = *a* \* *b*;

*iloraz* = *a* / *b*;

*potegowanie* = *a* \*\* 2;

Dane wyjściowe out.txt *out* := [*suma*, *roznica*, *iloczyn*, *iloraz*, *potegowanie*]

**Algorithm 1:** Działania algebraiczne na liczbach zespolonych

## Pełen kod aplikacji

```
1 @echo off
2 setlocal enabledelayedexpansion
3
4 :menu
5 echo =====
6 echo 1. Start
7 echo 2. Informacje
8 echo 3. Backup
9 echo 4. Zakoncz
10 echo =====
11
12 set /p input=Wybierz opcje:
13 if %input% EQU 1 goto start
14 if %input% EQU 2 goto info
15 if %input% EQU 3 goto backup
16 if %input% EQU 4 (
17     exit
18 ) else (
19     cls
20     goto menu
21 )
22
23 :start
24 call app.py
25 call web.py
26 cls
27 goto menu
28
29 :info
30 echo Program wykonuje podstawowe dzialania arytmetyczne w dziedzinie
    liczb zespolonych
31 echo Program pobiera dane wejsciowe z pliku
32 echo a nastepnie zwraca wynik w postaci strony
33 echo Program jest zadaniem konkursowym z konkursu Algorytmion 2017
34 echo Autor: Patryk Sroczynski
35 set /p c=Nacisnij enter
36 cls
37 goto menu
38
39 :backup
40 if not exist backup%date% (
41     mkdir backup%date%
42     copy *.* backup%date% > nul
43     echo Backup zapisany w folderze backup%date%
44 ) else (
45     copy *.* backup%date% > nul
46     echo Backup juz istnieje, pliki zostaly nadpisane
47 )
48 set /p c=Nacisnij enter
49 cls
50 goto menu
51 pause
```

---

```

1 import math
2
3 class Complex(object):
4
5     def __init__(self, real, imag = 0.0):
6         self.real = real
7         self.imag = imag
8
9     def __add__(self, other):
10         return Complex(self.real + other.real, self.imag + other.imag)
11
12     def __sub__(self, other):
13         return Complex(self.real - other.real, self.imag - other.imag)
14
15     def __mul__(self, other):
16         return Complex((self.real * other.real) - (self.imag * other.
17             imag),
18             (self.imag * other.real) + (self.real * other.imag))
19
20     def __truediv__(self, other):
21         m = other.real ** 2 + other.imag ** 2
22         return Complex((self.real * other.real + self.imag * other.imag)
23             / m,
24             (self.imag * other.real - self.real * other.imag) / m)
25
26     def __pow__(self, n):
27         r = pow(math.hypot(self.real, self.imag), n)
28         phi = n * ((math.pi / 2 - math.atan2(self.real, self.imag)) % (
29             math.pi * 2))
30         return Complex(math.cos(phi) * r, math.sin(phi) * r)
31
32 numbers = []
33
34 try:
35     file = open("in.txt", "r")
36     for line in file.readlines():
37         try:
38             numbers.append(float(line))
39         except ValueError:
40             numbers.append(1)
41     file.close()
42
43 try:
44     x1 = numbers[0]
45 except IndexError:
46     x1 = 1
47
48 try:
49     i1 = numbers[1]
50 except IndexError:
51     i1 = 1
52
53 try:
54     x2 = numbers[2]
55 except IndexError:
56     x2 = 1
57
58 try:
59     i2 = numbers[3]

```

```

53     except IndexError:
54         i2 = 1
55     try:
56         n = numbers[4]
57     except IndexError:
58         n = 1
59
60     a = Complex(x1, i1)
61     b = Complex(x2, i2)
62     sum = a + b
63     dif = a - b
64     prod = a * b
65     quo = a / b
66     pow = Complex(3, 2)**3
67
68 except IOError:
69     print("Bład otwierania pliku")
70
71 file = open("out.txt", "w")
72 data = str(sum.real) + "\n" + str(sum.imag) + "\n" + str(dif.real) + "\n"
73       " + str(dif.imag) + "\n" + str(prod.real) + "\n" + str(prod.imag) + "
74       \n" + str(quo.real) + "\n" + str(quo.imag) + "\n" + str(pow.real) + "
75       \n" + str(pow.imag)
76 file.write(data)
77 file.close()

```

---



```

1 import webbrowser
2
3 # in.txt
4 numbers = []
5 try:
6     file = open("in.txt", "r")
7     for line in file.readlines():
8         try:
9             numbers.append(float(line))
10        except ValueError:
11            numbers.append(1)
12    file.close()
13    try:
14        x1 = numbers[0]
15    except IndexError:
16        x1 = 1
17    try:
18        i1 = numbers[1]
19    except IndexError:
20        i1 = 1
21    try:
22        x2 = numbers[2]
23    except IndexError:
24        x2 = 1
25    try:
26        i2 = numbers[3]
27    except IndexError:
28        i2 = 1
29    try:
30        n = numbers[4]
31    except IndexError:
32        n = 1
33 except IOError:
34     print("Blad otwierania pliku")
35
36 #out.txt
37 results = []
38 try:
39     file = open("out.txt", "r")
40     for line in file.readlines():
41         try:
42             results.append(float(line))
43         except ValueError:
44             results.append(0)
45     file.close()
46 except IOError:
47     print("Blad otwierania pliku")
48
49 if len(results) < 9:
50     results = [0, 0, 0, 0, 0, 0, 0, 0, 0]
51
52 def plusorminus(number):
53     if number >= 0:
54         return " + " + str(number)
55     else:

```

```

56         return " " + str(number)
57
58     try:
59         file = open("index.html", "w", encoding="utf-8")
60         content = """
61 <!DOCTYPE html>
62 <html lang="pl">
63 <head>
64     <meta charset="UTF-8">
65     <meta http-equiv="X-UA-Compatible" content="IE=edge">
66     <meta name="viewport" content="width=device-width, initial-scale=1.0
67     ">
68     <title>Liczby zespolone</title>
69     <style>
70         *, body {
71             margin: 0;
72             padding: 0;
73             color: rgb(255, 255, 255);
74             font-size: 24px;
75         }
76         table {
77             width: 50vw;
78             height: 50vh;
79             margin-top: 10vh;
80         }
81         table,
82         td {
83             border: 3px solid #333;
84             text-align: center;
85         }
86         table, th {
87             background: rgb(114, 114, 245);
88         }
89
90         thead,
91         tfoot {
92             background-color: #333;
93             color: #fff;
94         }
95         .center {
96             margin-left: auto;
97             margin-right: auto;
98         }
99     </style>
100 </head>
101 <body>
102     <table class="center">
103         <thead>
104             <tr>
105                 <th colspan="4">Dzia[U+FFFD]nia na liczbach zespolonych</th>
106             </tr>
107         </thead>
108         <tbody>
109             <tr>
110                 <td>Dodawanie</td>

```

```

110         <td>"" + str(x1) + plusorminus(i1) + ""i</td>
111         <td>"" + str(x2) + plusorminus(i2) + ""i</td>
112         <td>"" + str(round(results[0])) + " " + plusorminus(
            round(results[1])) + ""i</td>
113     </tr>
114     <tr>
115         <td>Odejmowanie</td>
116         <td>"" + str(x1) + plusorminus(i1) + ""i</td>
117         <td>"" + str(x2) + plusorminus(i2) + ""i</td>
118         <td>"" + str(round(results[2])) + " " + plusorminus(
            round(results[3])) + ""i</td>
119     </tr>
120     <tr>
121         <td>Mnozenie</td>
122         <td>"" + str(x1) + plusorminus(i1) + ""i</td>
123         <td>"" + str(x2) + plusorminus(i2) + ""i</td>
124         <td>"" + str(round(results[4])) + " " + plusorminus(
            round(results[5])) + ""i</td>
125     </tr>
126     <tr>
127         <td>Dzielenie</td>
128         <td>"" + str(x1) + plusorminus(i1) + ""i</td>
129         <td>"" + str(x2) + plusorminus(i2) + ""i</td>
130         <td>"" + str(round(results[6])) + " " + plusorminus(
            round(results[7])) + ""i</td>
131     </tr>
132     <tr>
133         <td>Potegowanie</td>
134         <td>"" + str(x1) + plusorminus(i1) + ""i</td>
135         <td> </td>
136         <td>"" + str(round(results[8])) + " " + plusorminus(
            round(results[9])) + ""i</td>
137     </tr>
138 </tbody>
139 </table>
140 </body>
141 </html>
142 ""
143     file.write(content)
144     file.close()
145     webbrowser.open_new_tab('index.html')
146
147 except IOError:
148     print("Blad otwierania pliku")

```

---

Koniec i bomba. A kto czytał ten trąba. W. G.