1

Programowanie obiektowe

Opracował: L. Jung

Liczby zespolone

Początki liczb zespolonych, to XVI wiek, kiedy zostały one wprowadzone do algebry. Zaczął ich używać Rafael Bombelli, który nie był matematykiem. Był on inżynierem kierującym pracami przy osuszaniu bagien i terenów błotnych w Toskanii.

Obecnie liczby zespolone są stosowane zarówno w matematyce, fizyce, grafice (np. kreślenie fraktali, np. zbiorów Mandelbrota i Julii), elektronice (obliczanie obwodów prądu sinusoidalnego), automatyce, aerodynamice, , optyce, telekomunikacji, chemii czy też medycynie.

Postać algebraiczna (kanoniczna) liczb zespolonych

Liczba zespolona jest parą uporządkowanych liczb rzeczywistych (a,b), które w tzw. postaci algebraicznej (kanonicznej) zapisujemy następująco:

$$z = a + i*b$$

gdzie:

- **a** nazywana jest częścią rzeczywistą, którą oznaczamy jako ***Re z***,
- **b** nazywana jest częścią urojoną, którą oznaczamy jako ***Im z***,
- **i** nazywane jest *imaginarius (jednostka urojona):* oznacza ona pierwiastek **z** liczby **-1**.

Oznaczenia:

$$i=\sqrt{-1} \qquad i^2=-1$$

Właściwości liczby zespolonej

1) Liczba zespolona jest równa zero, wtedy i tylko wtedy, gdy `a = 0` i `b = 0`.
2) Jeżeli w liczbie zespolonej:

$$z = a + i*b$$

   - częścią urojoną `b = Im z` jest równa zero, to `z=a` , czyli liczba zespolona jest liczbą rzeczywistą,
   - częścią rzeczywista `a = Re z` jest równa zero, to `z= i*b` , czyli liczba zespolona jest liczbą urojoną.

3) Modułem liczby zespolonej $z = a + i*b$ nazywamy liczbę:

$$|z|=\sqrt{a^2+b^2}$$

4) Liczby zespolone są równe, gdy mają jednakowe zarówno części rzeczywiste i części urojone:

$$z_1 = z_2 \Leftrightarrow Re(z_1) = Re(z_2) \wedge Im(z_1) = Im(z_2)$$

Opracował: L. Jung

Działania na liczbach zespolonych

Dodawanie liczb zespolonych:

$$z_1 + z_2 = (a_1 + ib_1) + (a_2 + ib_2) = (a_1 + a_2) + i(b_1 + b_2)$$

Odejmowanie liczb zespolonych:

$$z1 - z2 = (a1 + ib1) - (a2 + ib2) = (a1 - a2) + i(b1 - b2)$$

Mnożenie liczb zespolonych:

$$z_1 * z_2 = (a_1 + ib_1)*(a_2 + ib_2) = (a_1a_2 - b_1b_2) + i\,(a_1b_2 + a_2b_1)$$

Dzielenie liczb zespolonych:

$$\frac{z_1}{z_2} = \frac{(a_1 + ib_1)}{(a_2 + ib_2)} = \frac{a_1a_2 + b_1b_2}{a_2^2 + b_2^2} + i\frac{b_1a_2 - a_1b_2}{a_2^2 + b_2^2}$$

Inne postacie zapisu liczb zespolonych

Wyróżnia się trzy rodzaje zapisu liczb zespolonych:

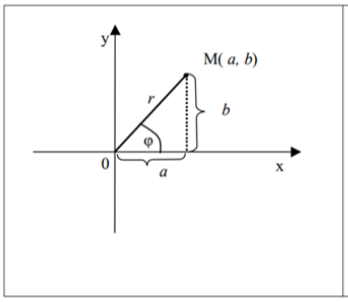
- algebraiczna (kanoniczna, tak jak powyżej),
- trygonometryczna (zwana też `polarną`),
- wykładnicza.

A wybór odpowiedniej postaci zapisu liczb zespolonych zależy, między innymi, od rodzaju wykonywanych obliczeń.

Opracował: L. Jung

Liczba zespolona w postaci trygonometrycznej (polarnej, biegunowej)

Liczbę zespoloną z = a + i*b możemy przedstawić na wykresie, podając odległość r punktu M(a, b) od początku układu współrzędnych oraz kąt ϕ jaki tworzy wektor OM r z dodatnim kierunkiem osi Ox:



Związki między zapisem kanonicznymi liczby zespolonej a jej zapisem w postaci trygonometrycznej:

$z = a + i * b =$

$r(\cos(\varphi) + i * \sin(\varphi))$

gdzie:

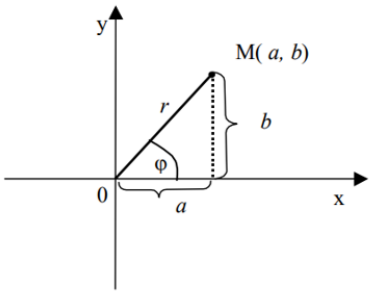
$a = r\cos(\varphi)$

$b = r\sin(\varphi)$

$r = \sqrt{a^2 + b^2}$

gdzie kąt φ nazywany jest argumentem liczby zespolonej

Argument liczby zespolonej φ jest miarą kąta skierowanego między wektorem reprezentującym liczbę zespoloną $z = a + i*b$ na płaszczyźnie zespolonej a osią rzeczywistą.



gdzie r =|z| i jest modułem liczby zespolonej:

$$r = \sqrt{a^2 + b^2}$$

Argument φ liczby zespolonej obliczamy z następującej zależności:

$$\varphi = \begin{cases} \operatorname{arctg}\left(\frac{b}{a}\right), & \text{gdy } a > 0 \\ \operatorname{arctg}\left(\frac{b}{a}\right) + \pi, & \text{gdy } a < 0 \\ 0, & \text{gdy } a = 0 \, oraz \, b > 0 \\ \pi, & \text{gdy } a = 0 \, oraz \, b < 0 \end{cases}$$

Argumentem liczby zespolonej $z = a + bi \neq 0$ nazywamy każdą liczbę rzeczywistą φ spełniającą dwa warunki:

$$cos(\varphi) = \frac{a}{|z|} \qquad sin(\varphi) = \frac{b}{|z|}$$

Argument φ liczby zespolonej z oznacza się $\varphi = \arg z$.

Układ współrzędnych, w którym podajemy współrzędne modułu liczby zespolonej i jej argumentu: (|z|, ϕ) nazywamy **układem współrzędnych biegunowych**, a współrzędne - **współrzędnymi biegunowymi**.

Liczba zespolona zapisana we współrzędnych biegunowych przyjmuje postać trygonometryczną:

$$z = |z|(\cos\varphi + i\sin\varphi), \text{ gdzie } \varphi = \arg z.$$

- Postać trygonometryczna liczb zespolonych **ułatwia wykonywanie** operacji na liczba zespolonych:

$$z_1 = |z_1| * (cos(\varphi_1) + i * \sin(\varphi_1))$$
$$z_2 = |z_2| * (cos(\varphi_2) + i * \sin(\varphi_2))$$

- **operacja mnożenia:**

$$z_1 * z_2 = |z_1| * |z_2| * (cos(\varphi_1 + \varphi_2) + i * \sin(\varphi_1 + \varphi_2))$$

- **operacja dzielenia:**

$$z_1 / z_2 = |z_1| / |z_2| * (cos(\varphi_1 - \varphi_2) + i * \sin(\varphi_1 - \varphi_2))$$

gdzie:

- $|z_1|$ jest modułem liczby zespolonej z_1: $|z_1| = r_1 = \sqrt{a_1^2 + b_1^2}$
- $|z_{21}|$ jest modułem liczby zespolonej z_2: $|z_2| = r_2 = \sqrt{a_2^2 + b_2^2}$

- **operacja potęgowania:**

$$z^n = (|z| * (cos(\varphi) + i * sin(\varphi))^n = |z|^n * (cos(n * \varphi) + i * sin(n * \varphi))$$

- **operacja pierwiastkowania:**

$$\sqrt[n]{z} = \sqrt[n]{||z||(cos(\varphi) + i * sin(\varphi))} =$$

$$\sqrt[n]{|z|}\left(cos\frac{\varphi + 2k\pi}{n} + i * sin\frac{\varphi + 2k\pi}{n}\right), \quad dla\ k = 0, 1, 2, .., n-1$$

gdzie |z| jest modułem liczby zespolonej: $|z| = \sqrt{a^2 + b^2}$

Co jest łatwiej wykonać niż ze wzorów dla algebraicznej postaci liczb zespolonych!

Liczba zespolona w postaci wykładniczej

Korzystając z formuły Eulera:

$$e^{i\varphi} = cos(\varphi) = i * sin(\varphi)$$

liczbę zespoloną:

```
z = a + i * b = r (cos(ϕ) + i * sin(ϕ))
```

możemy przedstawić w postać wykładniczej:

$$z = r * e^{i\varphi}$$

gdzie r jest modułem liczby zespolonej:

$$r = |z| = \sqrt{a^2 + b^2}$$

W formule Eulera wykorzystuje ona tzw. liczbę Eulera, oznaczaną jako "e". ("e" w przybliżeniu = 2, 71).

Projekt nowego typu danych: LiczbyZespolone

Przypomnienie: przez typ danych rozumiemy:

zbiór wartości (reprezentowany przez dany typ danych)

+

zbiór dozwolonych działań

Na przykład, typ danych int:

- reprezentuje 32-bitową liczbę całkowitą (z zakresu wartości: -2,147,483,648 do 2,147,483,647) zapisywane w 4bajtowej strukturze danych,
- dostępne operacje arytmetyczne (+, -, *, /, . . .,++, −−), bitowo logiczne (& - and, | - or, ^ - xor), relacyjne (<, >, ...)

Wybór struktury danych dla przechowywania wartości nowego typu danych

Dla zdefiniowania nowego typ danych (w języku C#) wybieramy:

- klasę (class), która jest typem referencyjnym obiektów (łączących listę pól składowych, listy metod i właściwości):

```
public class NazwaReferencyjnegoTypuDanych
    { .   .   .    }
```

- lub strukturę (struct), która jest typem wartościowym (również łączących listę pól składowych, listy metod i właściwości)

```
public struct NazwaWartościowegoTypuDanych
    { .   .   .   }
```

Różnice między klasą a strukturą:

- klasy są typem referencyjnym a struktury typem wartościowym,
- klasa obsługuje mechanizmu dziedziczenia i pamięć przydzielana jest na stercie programu,
- struktura nie obsługuje mechanizmu dziedziczenia i pamięć dla wartości struktury jest przydzielana na stosie programu,
- struktury nie mogą mieć domyślnego konstruktora.

- struktury stosowane są do reprezentowania prostych typów danych, takich jak liczby czy wartości logiczne,
- klasy stosowane są do reprezentowania złożonych typów danych, takich jak tablice,
- obiekty struktury są bardziej efektywne (wydajne) niż klasy, ponieważ są przechowywane na stosie, a obiekty klasy na stercie,
- struktury nie mogą mieć deklarowanego jawnie konstruktora bezparametrowego,

- obiekty struktury są typami wartości i tworzymy je przy użyciu operatora new:

  NazwaWartościowegoTypuDanych X = new NazwaWartościowegoTypuDanych(parametry dla konstruktora)

- można utworzyć obiekt struktury (typu wartość) bez użyciu operatora new, ale przed jego użyciem wszystkim jego polom muszą być nadane wartości (muszą być zainicjowane), co oznacza, że pola (zmienne) struktury muszą być publiczne (z możliwości ustalania ich wartości):

```
struct  Książka
   { public string tytuł;
     public string autor;
     public int Sygnatura;
    }
```


```
// Deklaracja książek typu "Książka"
Książka  książka_1;
Książka   książka_2;
// Specyfikacja pierwszej książki
książka_1.tytuł = „Rzecz o języku C#";
książka_1.autor = "Imię Nazwisko";
książka_1. Sygnatura = 32;
// Specyfikacja drugiej książki
książka_2.tytuł = "Wielka rozprawa filozoficzna";
książka_2.autor = "Imię Nazwisko";
książka_2. Sygnatura = 1;
// Przykładowe wypisanie interesujących danych
Console.WriteLine("Tytuł 1  książki: {0}", książka_1.tytuł);
Console.WriteLine("Tytuł 2  książki: {0}", książka_2.tytuł);
Console.ReadKey();
```

- można jednak utworzyć obiekt struktury (typu wartość) bez użyciu operatora new, ale z domyślnym przypisaniem wartości wszystkim polom (zmiennym) struktury, w następujący sposób:

  ```
  NazwaWartościowegoTypuDanych Y =
                    default(NazwaWartościowegoTypuDanych);
  ```

  co spowoduje obiekt struktury (typu wartość) na stosie i przypisanie wartości domyślnych dla wszystkich pól (zmiennych) zadeklarowanych w strukturze obiektu,
- składowe struktury nie mogą być zadeklarowane jako abstrakcyjne, wirtualne lub chronione.

Deklaracja typu danych: Liczby zespolone

Dla obliczeń na liczbach zespolonych wybieramy strukturę (struct), która jest typem wartościowym.
Musimy teraz pamiętać, że:

1) w deklaracji struktury nie można podać definicji (deklaracji) konstruktora bezargumentowego (jest natomiast dostępna tylko domyślna definicja konstruktora bezargumentowego!), a w klasie można!
2) w strukturze nie można inicjalizować pól klasy (np. private int Nmax = 55;),
3) struktura nie może dziedziczyć właściwości po innej strukturze lub klasie,
4) struktura nie może być strukturą bazową dla innych struktur lub klas,
5) do obiektu struktury odwołujemy się bezpośrednio jako do wartości, a do obiektów klasy odwołujemy się przez referencję (do egzemplarza klasy),

Projektujemy strukturę (struct) LiczbyZespolone, w której deklarujemy:

- pola (zmienne) dla przechowania wartości części rzeczywistej i urojonej liczby zespolonej:

```
public struct LiczbaZespolona
 {   // część rzeczywista liczby zespolonej
    private double re;
   // część urojona liczby zespolonej
    private double im;
```

A teraz deklarujemy:

- właściwości Liczby zespolonej (odczytywanie wartość rzeczywistej i urojonej oraz odczytywanie wartości **Normy** liczby zespolonej),
- indeksatora Liczb zespolonych,
- konstruktory (struktury `LiczbyZespolone`),
- przeciążanie operatorów jednoargumentowych:
  - + i -
  - ~ , który wyznacza liczbę sprzężoną podanej liczby zespolonej)
  - ! (wyznaczającego moduł (normę) liczby zespolonej):
- metode wyznaczającą `Normę` liczby zespolonej,
- przeciążanie operatorów dwuargumentowych (+, -, *, /, ==, !=, a także takich operatorów jak: `operację potęgowania`, `operacja pierwiastkowania`),
- przeciążanie metody `Equals` i `GetHashCode`,
- nadpisanie metody `ToString()` dokonującej konwersji liczby zespolonej na jej zapis algebraiczny (kanoniczny)
- operatora `implicite` konwersji liczby rzeczywistej na liczbę zespoloną,
- . . .
-

Szablon projektu obsługi zdarzenia `Click` dla przycisku poleceń:

`C = A + B` (o nazwie `btnSumaAB`):

```
private void btnSumaAB_Click(object sender, EventArgs e)
  {
   errorProvider1.Dispose(); // wyłączenie kontrolki errorProvider1
   // sprawdzenie, czy podane zostały liczby zespolone A  oraz B

    ……

    ……

    ……

  /* utworzenie egzemplarza obiektu (struktury) dla danych  liczb
     zespolonych: A i B  */
  LiczbaZespolona A = new LiczbaZespolona(..., . . .);
  LiczbaZespolona B = new LiczbaZespolona(..., . . .);
  /* ustawienie stanu braku aktywności dla kontrolek (typu textBox) z
których pobrano wartości danych wejściowych (liczb zespolonych: A i B)  */
    ……

    ……

    ……

  /* utworzenie egzemplarza obiektu (struktury) dla wyniku obliczeń:
  C, której część rzeczywista Re i urojona Im będzie wynikiem
  obliczenia sumy liczb zespolonych  */
  LiczbaZespolona C = A + B;

  // wizualizacja wyniku obliczeń
  txtWynik.Text = C.ToString();
  // metoda ToString() została nadpisana w strukturze
  LiczbyZespolone i umożliwia wypisanie liczby zespolonej w postaci
  kanonicznej

  // ustawienie stanu braku aktywności kontrolki z wynikiem
  txtWynik.Enabled = false;
 }
```