C++17 i STL

Liczby zespolone

Liczby zespolone

- Liczby zespolone składają się z dwóch części — rzeczywistej oraz urojonej.
- Część urojona posiada tę właściwość, iż podniesiona do kwadratu daje w wyniku liczbę ujemną; część urojoną liczby zespolonej stanowi współczynnik i, będący pierwiastkiem kwadratowym liczby –1.
- STL w C++ udostępnia szablon klasy complex<> w pliku nagłówkowym **<complex>**, umożliwiający wykonywanie obliczeń na liczbach zespolonych.

Liczby zespolone

- Klasa complex<> jest zdefiniowana w następujący sposób: template <typename T> class complex;
- Parametr szablonu o nazwie T używany jest jako typ skalarny zarówno dla części rzeczywistej, jak też urojonej danej liczby zespolonej.
- STL udostępnia trzy specjalizacje klsy comlex<>:
  - comlex<float>,
  - comlex<double>,
  - comlex<long double>.

Tworzenie liczb zespolonych

- Konstruktory umożliwiają przekazanie wartości początkowej poprzez określenie jej części rzeczywistej oraz urojonej. W przypadku gdy nie zostaną one określone, nastąpi ich zainicjalizowanie zerem określonego typu.
- // liczba zespolona z częścią rzeczywistą oraz urojoną
  // - część rzeczywista: 4.0
  // - część urojona: 3.0
  complex<double> c1(4.0, 3.0);
- // liczba zespolona utworzona przy wykorzystaniu współrzędnych biegunowych
  // - moduł: 5.0
  // - kąt fazy: 0.75
  complex<float> c2(polar(5.0,0.75));

Tworzenie, kopiowanie przypisanie do liczby zespolonej

- Operatory przypisania stanowią jedyny sposób modyfikacji wartości istniejącej liczby zespolonej.
- Przypisania liczb zespolonych (z ewentualną operacją arytmetyczną):
  c1 = c2
  c1 += c2
  c1 -= c2
  c1 *= c2
  c1 /= c2

Tworzenie, kopiowanie przypisanie do liczby zespolonej

- Funkcja pomocnicza o nazwie polar() umożliwia utworzenie liczby zespolonej, zainicjalizowanej za pomocą współrzędnych biegunowych (modułu oraz kąta fazy podanego w radianach). Przykład:
  complex<double> c2(std::polar(4.2, 0.75));
- Funkcja pomocnicza o nazwie conj() umożliwia utworzenie liczby zespolonej zainicjalizowanej za pomocą wartości sprzężonej z inną liczbą zespoloną (wartość sprzężona z daną liczbą zespoloną tworzona jest poprzez zanegowanie jej części urojonej):
  complex<double> c1(1.1, 5.5);
  complex<double> c2(conj(c1));

Funkcje związane z dostępem do wartości liczby zespolonej

- Część rzeczywista i urojona liczby zespolonej c:
  c.real(), real(c)
  c.imag(), imag(c)
- Moduł liczby zespolonej:
  abs(c)
- Norma liczby zespolonej (kwadrat modułu):
  norm(c)
- Kąt fazy liczby zespolonej:
  arg(c)
- Liczba sprzężona z liczbą zespoloną:
  conj(c)

Arytmetyka liczb zespolonych

- Operacje arytmetyczne na liczbach zespolonych c1 i c2:
  - c1 – negacja
  c1 + c2 – suma
  c1 - c2 – różnica
  c1 * c2 – iloczyn
  c1 / c2 – iloraz

Porównywanie liczb zespolonych

- W celu porównania dwóch liczb zespolonych możesz jedynie sprawdzić, czy są one równe.
- Operatory == oraz != zostały zdefiniowane jako funkcje globalne, dlatego też jeden z argumentów może być wartością skalarną.
- Jeśli w charakterze argumentu użyjesz skalara, jest on interpretowany jako część rzeczywista, zaś część urojona posiada wartość 0.
- Inne operacje porównania (operatory <, <=, > i >=) nie są zdefiniowane. W konsekwencji niemożliwe jest użycie liczby zespolonej typu complex<> jako elementu kontenera asocjacyjnego.

Operatory wejścia-wyjścia dla liczb zespolonych

- Klasa complex<> udostępnia powszechnie stosowane operatory wejścia-wyjścia << oraz >>.
- Operacje strumieniowe:
  strm << c
  strm >> c
- Operator wyjściowy zapisuje liczbę zespoloną przy uwzględnieniu stanu bieżącego strumienia w postaci pary:
  (część_rzeczywista, część_urojona)
- Operator wejściowy umożliwia odczytanie liczb zespolonych zapisanych w jednej z poniższych postaci:
  (część_rzeczywista, część_urojona)
  (część_rzeczywista)
  część_rzeczywista

Funkcje trygonometryczne i wykładnicze

- Funkcje trygonometryczne dla liczb zespolonych:
  sin(c) – sinus
  cos(c) – cosinus
  tan(c) – tangens
  asin(c) – arcus sinus
  acos(c) – arcus cosinus
  atan(c) – arcus tangens
- Funkcje wykładnicze dla liczb zespolonych:
  pow(c1, c2) – potęgowanie
  sqrt(c) – pierwiastek
  exp(c) – funkcja eksponencjalna
  log(c) – logarytm naturalny

Algorytmy numeryczne

- Aby móc użyć algorytmów numerycznych, należy dołączyć plik nagłówkowy <numeric>.
- Za pomocą algorytmów numerycznych można przetwarzać nie tylko pojedyncze wartości numeryczne ale także całe sekwencje tych wartości.
- Założenie: elementy numeryczne są zgromadzone w kolekcji sekwencyjnej.

Algorytmy numeryczne – obliczanie wartości wypadkowej ciągu

- Obliczanie wartości wypadkowej jednego ciągu:
  T accumulate (InputIterator beg, InputIterator end, T initValue)
  T accumulate (InputIterator beg, InputIterator end,
  T initValue, BinaryFunc op)
- Pierwsza postać algorytmu oblicza i zwraca sumę wartości initValue oraz wszystkich elementów z zakresu [beg,end). W szczególności, dla każdego elementu wykonuje ona operację:
  initValue = initValue + elem
- Druga postać algorytmu oblicza i zwraca wynik wywołania operacji op wobec wartości initValue oraz wszystkich elementów z zakresu [beg,end). W szczególności, dla każdego elementu wykonuje ona operację:
  initValue = op(initValue, elem)

Algorytmy numeryczne – obliczanie iloczynu skalarnego dwóch ciągów

- Obliczanie iloczynu skalarnego dwóch ciągów:
  T inner_product (InputIterator beg1, InputIterator end1,
  InputIterator beg2, T initValue)
  T inner_product (InputIterator beg1, InputIterator end1,
  InputIterator beg2, T initValue ,
  BinaryFunc op1 , BinaryFunc op2)
- Pierwsza postać algorytmu oblicza i zwraca iloczyn skalarny wartości initValue oraz wszystkich elementów z zakresu [beg1,end1) w kombinacji z elementami z zakresu rozpoczynającego się od pozycji beg2 . W szczególności, dla każdej pary elementów wykonuje ona operację:
  initValue = initValue + elem1 * elem2
- Druga postać algorytmu dla każdej pary elementów wykonuje operację:
  initValue = op1(initValue, op2(elem1, elem2))