Zadanie nr 2: Arytmetyka liczb zespolonych

1 Cel ćwiczenia

Wykształcenie umiejętności definiowania przeciążeń operatorów arytmetycznych dwuargumentowych i jednoargumentowych dla własnych struktur danych oraz operatora porównania. Przećwiczenie podstawowych operacji na strumieniach plikowych. Nabycie umiejętności definiowania przeciążeń operacji na strumieniach.

2 Opis zadania programowego

Należy napisać program, który umożliwia wykonanie prostego sprawdzianu z arytmetyki liczb zespolonych. Sprawdzian ma dotyczyć operacji dodawania, odejmowania, mnożenia i dzielenia. Zakłada się, że są dostępne co najmniej dwa zestawy pytań. Dla uproszczenia zakładamy, że liczby zespolone będą wyświetlane w nawiasach. W tej formie będą również wprowadzane,
Przykładowy zapis wyrażeń podany jest poniżej.

```
(4.2+2i)+(0+2i)
(3-0i)*(-1+2.1i)
(0-4i)/(3-1i)
```

W zapisie tym dla ułatwienia zachowanych jest kilka reguł:

- każda liczba zespolona zawsze zapisywana jest w nawiasach półokrągłych,
- zawsze występuje liczba oznaczająca część rzeczywistą, niezależnie od tego czy jest ona różna od zera, czy też równa jest zeru,
- zawsze występuje liczba oznaczająca część urojoną, niezależnie od tego czy jest ona różna od zera, czy też równa jest zeru,
- zestaw pytań zakończony jest znakiem kropki.

W trakcie realizacji testu program wyświetla kolejne wyrażenie i pyta użytkownika o wynik operacji. Udzieloną odpowiedź sprawdza z poprawnym wynikiem. W przypadku poprawnej odpowiedzi, potwierdza ten fakt, zaś w przypadku błędnej odpowiedzi stwierdza ten fakt i podaje odpowiedź właściwą. Po zakończeniu testu podawana jest statystyka błędnych i poprawnych odpowiedzi.

2.1 Opis działania programu

Program ma być uruchomiony z parametrem wskazującym wariant testu. Zakłada się, że wystąpią co najmniej dwa warianty testu o nazwie `latwy` i `trudny`. Nazwa ta musi się pojawić w linii wywołania programu jako opcja, np.

```
./test_arytmetyki_zespolonej latwe
```

Po uruchomieniu, program przechodzi od razu w tryb zadawania pytań. W przypadku błędnego wprowadzenia liczby ma dać użytkownikowi 3 razy szansę wprowadzenia liczby we właściwym formacie. Jeśli to się nie uda, odpowiedź należy uznać jako błędną. Po zakończeniu testu i wyświetleniu statystki odpowiedzi, program kończy swoje działanie.

2.2 Działanie programu

Układ pytań i odpowiedzi w niniejszym przykładzie należy traktować jako obowiązujący. Szczegółowe elementy, takie jak *minki*, są jedynie propozycją.

```
jkowalsk@noxon: rozwiazanie> ./test_arytmetyka latwy

:? Podaj wynik operacji:  (1.2+2i) + (0+2i) =
   Twoja odpowiedz: (1.2+3i)
:( Blad. Prawidlowym wynikiem jest: 1.2+4i

:? Podaj wynik operacji:  (2-0i) * (-1+2.3i) =
   Twoja odpowiedz: (-z+4.6i)

  Blad zapisu liczby zespolonej. Sprobuj jeszcze raz.

   Twoja odpowiedz: (-2+4.6i)
:) Odpowiedz poprawna

:? Podaj wynik operacji: (1-4i) / (0-2i)  =
   Twoja odpowiedz: (2+0.5i)
:) Odpowiedz poprawna

:? Podaj wynik operacji: (1+2i) - (1-2i)
   Twoja odpowiedz: (0+4i)
:) Odpowiedz poprawna


:? Podaj wynik operacji: (0+0i) * (1241-2890i)
   Twoja odpowiedz: (0+0i)
:) Odpowiedz poprawna

Koniec testu.

Ilosc dobrych  odpowiedzi: 3
Ilosc blednych odpowiedzi: 1
Wynik procentowy poprawnych odpowiedzi:    75.0%

jkowalsk@noxon: rozwiazanie>_
```

2.3 Definicje działań arytmetycznych

Implementując działanie dzielenia liczb zespolonych należy wziąć pod uwagę, że można je zapisać następująco

$$\frac{z_1}{z_2} = \frac{z_1\bar{z}_2}{z_2\bar{z}_2} = \frac{z_1\bar{z}_2}{|\, z_2 \,|^2} \tag{1}$$

gdzie $\bar{z}_2$ to liczba zespolona sprzężona do z_2, zaś $|\, z_2 \,|$ jest modułem tej liczby. Należy zwrócić uwagę, że w ostatnim przypadku mamy dzielenie liczby zespolonej przez liczbę rzeczywistą. Oznacza to, że należy zaimplementować ten typ operacji.

2.4 Wymagania co do konstrukcji programu

Oprócz wymagań sformułowanych w opisie zadania należy uwzględnić uwarunkowania przedstawione poniżej.

- Program musi zachować strukturę modułową i odpowiednią strukturę kartotek. O ile będzie to konieczne, należy zmodyfikować plik `Makefile` (np. gdy dodany zostanie nowy moduł).
- Program powinien mieć osobny moduł, który zawiera definicje wszystkich niezbędnych funkcji dla struktury `LZespolona` oraz przeciążeń działań: dodawania, odejmowania, mnożenia i dzielenia. Musi on również zawierać przeciążenia operacji na strumieniach dla tej klasy. Należy również zdefiniować dzielenie przez liczbę i porównanie z liczbą. Nagłówek do tego modułu powinien zawierać definicję tej klasy i zapowiedź definicji odpowiednich przeciążeń.
- Operacje wyświetlania, jak też wczytywania liczb zespolonych (zarówno ze strumienia standardowego jak też plikowego) w finalnej wersji programu muszą być realizowane **wyłącznie** z wykorzystanie przeciążeń operatorów odpowiednio:
  - przesunięcia bitowego w lewo (`<<`) dla strumienia wyjściowego (`cout`),
  - przesunięcia bitowego w prawo (`>>`) dla strumienia wejściowego (`cin`).
- Program **nie może** operować za pomocą funkcji języka C, takich jak `scanf`, `fscanf`, `printf`, `fprintf` itd. na standardowych strumieniach (jak też strumieniach plikowych).
- Należy też w sposób przemyślany wydzielić te funkcje, które biorą udział w interakcji z użytkownikiem i generują odpowiednie komunikaty dla niego, oraz te, które tego nie robią. W szczególności w bezpośredniej interakcji nie powinny brać udział przeciążenia operatorów. Wyjątkiem są tu komunikaty o wystąpieniu błędu fatalnego. W takim przypadku komunikat musi być kierowany na wyjście *standard error* (dostęp do niego jest poprzez obiekt `cerr`), a nie *standard output* (dostęp do niego jest poprzez obiekt `cout`).
- Wszystkie funkcje należy opisać. Należy także zwrócić uwagę, że szczególnie w przypadku operacji dzielenia niezbędne są odpowiednie warunki wstępne (jakie?).

W trakcie kompilacji finalnej wersji programu nie mogą generować się żadne ostrzeżenia. Kompilację należy wykonywać z opcjami `-Wall` oraz `-pedantic`. Niniejsza uwaga dotyczy tego i wszystkich następnych programów realizowanych w ramach niniejszego kursu, o ile w opisie zadania nie będzie innych zaleceń.

3 Pomoc – zalążek

W katalogu `~bk/edu/kpo/zad/z2/zalazek` znajduje się zalążek programu wraz ze wstępnie przygotowanymi modułami. Zawiera on również plik `Makefile`, który pozwala przeprowadzić kompilację i konsolidację programu, a po zakończeniu jej sukcesem, uruchamia program. Zalążek ten nie zawiera modułu, w którym byłyby definicje dla struktury danych niezbędnej na potrzeby obliczania statystyki. Należy go dodać we własnym zakresie, jako ćwiczenie pozwalające nabyć podstawowe umiejętności modyfikacji pliku `Makefile`. Struktura dostarczonego zalążka ma postać:

```
.
|
+-- Makefile
+-- inc
|    +-- BazaTestu.hh
|    +-- LZespolona.hh
|    +-- WyrazenieZesp.hh
+-- src
     +-- BazaTestu.cpp
     +-- LZespolona.cpp
     +-- main.cpp
     +-- WyrazenieZesp.cpp
```

Konstrukcja struktury `BazaTestu` bazuje na tablicach statycznych w module `BazaTestu.cpp`, które zawierają wcześniej przygotowany zestaw wyrażeń zespolonych dla testu w wersji łatwej i wersji trudnej. Definicja w tym module jest jedynie zaczątkiem i należy zawartość tego modułu rozszerzyć.

4 Rozszerzenia

W dalszej części opisane jest rozszerzenie zakresu zadania. Jej realizacja nie jest obowiązkowa. Przeznaczona one jest dla tych osób, które dobrze sobie radzą z programowaniem i pragnęłyby uzyskać wyższe oceny.

4.1 Czytanie skróconej notacji

Operacja czytania liczby zespolonej powinna dawać poprawną interpretację skróconej notacji, która pomija liczbę zero oraz jedynkę. Tak więc taki program powinien umieć odczytać następujące wyrażenia:

```
(1.2+2i)+(2i)
(2)*(-1+2.3i)
(-4i)/(3-i)
```

4.2 Czytanie pytań z pliku

Należy przebudować moduł `BazaTestu`, tak aby zadania testowe były czytane z pliku. Wszystkie operacje związane z czytaniem pytań należy wykonywać na strumieniach plikowych z wykorzystaniem mechanizmów dostępnych w języku C++. Wywołując program z opcją `latwe` należy otworzyć i czytać pytania z pliku `latwe.dat`. Analogicznie w przypadku użycia opcji `trudne`. Czytając zawartość pliku należy kontrolować poprawność czytanych danych oraz także kontrolę poprawności otwarcia pliku. W przypadku napotkania błędnego zapisu wyrażenia w pliku (może to się odnosić do błędu liczby, formy nawiasu lub operatora) program powinien poinformować o wykrytym błędzie, pominąć tę linię i przejść do czytania następnego wyrażenia. Komunikat taki powinien mieć formę.

```
Napotkano bledne wyrazenie. Zostalo ono pominiete.
```

Tego typu komunikaty powinny być wyświetlane na wyjściu `standard error` (`cerr`).

Poniżej przedstawiona jest przykładowa zawartość pliku. Zawiera ona specjalnie wprowadzone błędne wyrażenia.

```
(1.2+2i) + (0+2i)
(2-0i) * (-1+1.3i)
(2-0i) & (-1+1.3i)
(1-4i) / (0-2i)
(C-0i) + (-1+1.3i)
(1+2i) - (1-2i)
[1-0i] * [-1+1.3i]
(0+0i) * (1241-2890i)
```

Przykładowe pliki dostępne są w katalogu `~bk/edu/kpo/zad/z2/pliki_testow`.

5 Wymagania i zarys programu zajęć w okresie realizacji zadania

Przystępując do zajęć należy zapoznać się i wykorzystać zalążek znajdujący się w kartotece `~bk/edu/kpo/zad/z2/zalazek`.

5.1 Tydzień 0

Pod kierunkiem osoby prowadzącej należy dokonać analizy problemu pod względem niezbędnych struktur danych oraz algorytmu działania programu. We wspomnianym algorytmie należy zwrócić uwagę na operacje, które później będą musiały zostać zaimplementowane jako osobne funkcje. W szczególności trzeba określić jakie funkcje będą powiązane z jakimi strukturami danych. Skutkować to będzie umieszczeniem ich w odpowiednich modułach.

W dalszej części należy zdefiniować funkcję wyświetlającą wartość zmiennej zespolonej na wyjście standardowe. Proponowany prototyp tej funkcji to:

```
void Wyswietl(LZespolona);
```

W następnym kroku należy zdefiniować przeciążenia operatorów: $-$, $*$ oraz $/$; dla operacji, odpowiednio: odejmowania, mnożenia oraz dzielenia liczb zespolonych. Na potrzeby realizacji operacji dzielenia zalecane jest wykorzystanie wzoru (1) w podrozdziale 2.3. Należy przy tym zwrócić uwagę, że aby to było możliwe konieczne jest zdefiniowanie dwóch prostych funkcji dla typu `LZespolona`, które pozwalają wyznaczyć sprzężenie oraz kwadrat modułu liczby zespolonej. Proponowane prototypy tych funkcji to:

```
LZespolona Sprzezenie(LZespolona);
double Modul2(LZespolona);
```

Ponadto konieczne będzie zdefiniowanie przeciążenia operatora dzielenia dla wykonania operacji dzielenia zmiennej zespolonej przez liczbę typu `double`. Dla każdej definicji przeciążenia operacji arytmetyczny należy przeprowadzić prosty test jej działania modyfikując odpowiednio funkcję `main`. W dalszej części należy zdefiniować odpowiednie struktury danych dla statystyki odpowiedzi oraz utworzyć osobny moduł, w którym znajdować się będzie definicja funkcji umożliwiającej wyświetlenie statystyki. Moduł ten należy dołączyć do całości poprzez odpowiednie zmodyfikowanie pliku `Makefile` (schemat prostej konstrukcji pliku `Makefile` znajduje się w dodatkowej prezentacji powiązanej z tym zadaniem).

5.2 Tydzień 1

W ramach przygotowania do zajęć muszą zostać wcześniej napisane definicje następujących funkcji:

- wyświetlającej wyrażenie zespolone,
- interpelującej wyrażenie zespolone i obliczające jego wartość,
- funkcja wczytująca liczbę zespoloną,
- funkcja wczytująca wyrażenie zespolone,
- opis funkcji oraz opis przeciążeń dla operacji dzielenia.

Funkcje te należy wstępnie przetestować. Wszystko co będzie ponad to będzie oceniane *in plus* Natomiast w trakcie zajęć należy napisać przeciążenia operatorów przesunięcia bitowego w lewo (<<) dla wyświetlania liczby zespolonej i wyrażenia zespolonego. oraz przesunięcia bitowego w prawo (>>) dla czytania liczby zespolonej.

5.3 Tydzień 2

Rozliczenie się z gotowego programu i rozpoczęcie następnego zadania.