Zadanie 2: Arytmetyka symboli

1 Cel ćwiczenia

Wykształcenie umiejętności abstrahowania operacji arytmetycznych. Zapoznanie się i przećwiczenie mechanizmu tworzenia przeciążeń funkcji operatorowych. Utrwalenie umiejętności tworzenie opisów funkcji.

2 Program zajęć

Materiały pomocnicze do zajęć znajdują się w kartotece `~bk/edu/kpo/zad/z2`. Występujące w dalszej części opisu nazwy podkatalogów odnoszą się do zasobów występujących we wcześniej wspomnianej kartotece.

- *Ocena przygotowania do bieżących zajęć* – w ramach przygotowania do zajęć należy napisać kod definicji operacji dodawania symboli. Szczegółowy opis wymaganego przygotowania do zajęć znajduje się w podrozdziale 5. Ocenie podlegać będzie poprawność definicji i kompilacji pliku.

- *Realizacja wstępnej fazy prac nad nowym zadaniem* –

  W ramach wstępnej fazy pracy nad programem należy napisać funkcję realizującą wyliczanie elementu przeciwnego względem operacji dodawania. W oparciu o nią należy zdefiniować operację odejmowania (definicja funkcji i przeciążenie operatora `'-'`). Należy też w funkcji `main` przetestować wywołanie tej funkcji oraz realizacji operacji dodawania z wykorzystaniem wspomnianego operatora. Opis zadania znajduje się w podrozdziale 3. Stworzone funkcje i przeciążenia operatorów powinny zostać odpowiednio opisane.

- *Ocena realizacji wstępnej wersji programu*

3 Opis zadania programowego – przeciążanie operatorów

Należy oprogramować operacje arytmetyczne takie jak dodawanie, odejmowanie, mnożenie i dzielenie. Operacje te mają być realizowane na zbiorze symboli $S = \{e, a, b, c, d\}$ reprezentowanych przez wybrane litery alfabetu. W programie należy dokonać odpowiedniego *przeciążenia* operatorów $+, -, *, /$, aby zaimplementować wcześniej wymienione operacje. Należy je również zaimplementować za pomocą funkcji `Dodaj`, `PrzeciwnyDodawania`, `Odejmuj`, `Mnoz`, `OdwrotnyMnozenia`, `Dziel`.

W programie należy te same wyrażenie arytmetyczne podane przez prowadzącego zapisać w postaci:

1. zwyczajowego zapisu wyrażeń arytmetycznych (będzie to możliwe dzięki przeciążeniu operatorów),
2. wywołań funkcji,
3. jawnych wywołań przeciążonych funkcji operatorowych.

Wynik obliczeń powinien zostać porównany i wyświetlony komunikat, czy wyniki obliczeń są identyczne, czy też nie.

3.1 Operacja dodawania

Operacja dodawania zadana jest w postaci tabelki, która uwzględnia wszystkie możliwe kombinacje argumentów tej operacji. Ma ona następującą postać:

Działanie "+"

	e	a	b	c	d
e	e	a	b	c	d
a	a	d	e	b	c
b	b	e	c	d	a
c	c	b	d	a	e
d	d	c	a	e	b

Tabelkę tę należy czytać w następujący sposób

$$argument\_1_{wiersz} \quad operator \quad argument\_2_{kolumna} = element\_macierzy_{wiersz,kolumna}.$$

Na przykład dla operacji dodawania mamy

$$b + d = a.$$

Należy zauważyć, że elementem neutralnym tej operacji jest symbol e, tzn. niezależnie do jakiego symbolu dodamy symbol e wynikiem będzie pierwszy z symboli, np.

$$b + e = b.$$

Warto też zauważyć, że dla każdego elementu istnieje element przeciwny, których suma da element neutralny. Na przykład dla b takim elementem jest c, gdyż zgodnie z tabelką działania

$$b + a = e.$$

Zbiór S z tak zdefiniowanym działaniem dodawania tworzy strukturę algebraiczną $(S, +)$ zwaną *grupą*. Ze względu na to, że działanie to jest przemienne, to strukturę taką nazywamy *grupą przemienną* lub *grupą abelową*.

3.2 Operacja odejmowania

Bazując na wcześniej zdefiniowanej operacji dodawania, odejmowanie możemy zdefiniować jako operację dodawania elementu przeciwnego. Podstawową własnością elementu przeciwnego jest to, że dodając go do danego elementu otrzymamy element neutralny. Dla przykładu rozważmy element b i oznaczmy element przeciwny do niego jako $(-b)$. Mając na uwadze wcześniej opisaną własność mamy

$$b + (-b) = e.$$

Ze zdefiniowanej tabelki dodawania wynika, że elementem, który możemy wstawić w miejsce $(-b)$, gdyż spełnia przedstawiony warunek, jest a. Mamy więc

$$(-b) = a.$$

Tak więc przykładową operację odejmowania można zapisać jako

$$c - b = c + (-b) = c + a = b.$$

3.3 Operacja mnożenia

Operację mnożenia, podobnie jak operację dodawania, zdefiniujemy poprzez tabelkę (odpowiednik tabliczki mnożenia ;-)

Działanie "$*$"

	e	a	b	c	d
e	e	e	e	e	e
a	e	a	b	c	d
b	e	b	a	d	c
c	e	c	d	b	a
d	e	d	c	a	b

Można zauważyć, że w tym przypadku elementem neutralnym tej operacji (pełniącym rolę jedności) jest symbol a. A więc gdy przemnożymy dowolny symbol przez a zawsze otrzymamy ten sam symbol, np.

$$a * b = b.$$

To pozwala nam zdefiniować pojęcie symbolu odwrotnego do danego symbolu. Rozważmy symbol d. Symbol do niego odwrotny względem mnożenia oznaczmy jako d^{-1}. Tak więc po przemnożeniu tych dwóch symboli powinniśmy otrzymać symbol neutralny względem operacji mnożenia, a więc symbol a. Łatwo zauważyć, że w tym przypadku takim symbolem jest symbol c, a więc

$$c * d = a.$$

3.4 Operacja dzielenia

Operację dzielenia możemy rozumiemy analogicznie jak operację odejmowania. A więc operacja podzielenia b przez c odpowiada przemnożeniu b przez element odwrotny do c względem mnożenia. Oznaczmy taki element jako c^{-1}. Tak więc mamy

$$b/c = b * c^{-1} = b * d = c.$$

Uwaga: W przypadku operacji mnożenia nie wszystkie elementy mają swój element odwrotny (jaki to jest element w tym przypadku?). Należy to odpowiednio uwzględnić implementując operację wyszukiwania elementu odwrotnego oraz operacji dzielenia.

3.5 Materiały pomocnicze

W podkatalogu `oper` znajduje się plik `dzialania.cpp`. Jest on szkieletem programu, który należy uzupełnić tak, aby utworzyć program implementujący wymienione wcześniej działania. Ponadto dodatkowe materiały, które mogą być pomocne można znaleźć w dokumencie pod adresem:

`http://rab.ict.pwr.wroc.pl/~kreczmer/po/materialy/operatory-cz1.pdf`

3.6 Zawartość pliku `dzialania.cpp`

```
#include <iostream>

using namespace std;
```

```
/*
  Definicja typu wyliczeniowego "Symbol" dla symboli  a, b, c, d, e.
*/

/*
  Definicja funkcji: Dodaj, PrzeciwnyDodawania, Odejmij, Mnoz, OdwrotnyMnozenia, Dziel.
*/

/*
  Definicja przeciążeń operatorów
*/

void Porownaj(Symbol wynik_alg, Symbol wynik_ope, Symbol wynik_fun)
{
 cout << ((wynik_alg == wynik_fun) ? "Identyczne" : "Rozne")
      << " wyniki wyrazenia algebraicznego i funkcyjnego." << endl;

 cout << ((wynik_ope == wynik_fun) ? "Identyczne" : "Rozne")
      << " wyniki wyrazenia operatorowego i funkcyjnego." << endl;

 cout << ((wynik_ope == wynik_alg) ? "Identyczne" : "Rozne")
      << " wyniki wyrazenia operatorowego i algebraicznego." << endl;
}

int main()
{
 Symbol wynik_alg, wynik_ope, wynik_fun;

/*--------------------
  Po wpisaniu odpowiednich wyrażeń znaki komentujące
  poniższy zestaw instrukcji należy usunąć.

 wynik_alg = WYRAZENIE_ALG_1; // <- Wyrażenie zapisane w sposób zwyczajowy
 wynik_fun = WYRAZENIE_FUN_1; // <- Wyrażenie wykorzystujące funkcje
 wynik_ope = WYRAZENIE_OPE_1; // <- Wyrażenie z jawnym wywołaniem operatorów

 Porownaj(wynik_alg,wynik_ope,wynik_fun);

 wynik_alg = WYRAZENIE_ALG_2; // <- Wyrażenie zapisane w sposób zwyczajowy
 wynik_fun = WYRAZENIE_FUN_2; // <- Wyrażenie wykorzystujące funkcje
 wynik_ope = WYRAZENIE_OPE_2; // <- Wyrażenie z jawnym wywołaniem operatorów

 Porownaj(wynik_alg,wynik_ope,wynik_fun);

 wynik_alg = WYRAZENIE_ALG_3; // <- Wyrażenie zapisane w sposób zwyczajowy
 wynik_fun = WYRAZENIE_FUN_3; // <- Wyrażenie wykorzystujące funkcje
 wynik_ope = WYRAZENIE_OPE_3; // <- Wyrażenie z jawnym wywołaniem operatorów
```

```
 Porownaj(wynik_alg,wynik_ope,wynik_fun);

 wynik_alg = WYRAZENIE_ALG_4; // <- Wyrażenie zapisane w sposób zwyczajowy
 wynik_fun = WYRAZENIE_FUN_4; // <- Wyrażenie wykorzystujące funkcje
 wynik_ope = WYRAZENIE_OPE_4; // <- Wyrażenie z jawnym wywołaniem operatorów

 Porownaj(wynik_alg,wynik_ope,wynik_fun);
 --------------------*/
}
```

Przykład realizacji zapisu działań dla wyrażenia: `(e + a) * c`

```
 wynik_alg = (e + a) * c;
 wynik_fun = Mnoz(Dodaj(e,a),c);
 wynik_ope = operator * (operator + (e,a), c);
```

4 Wymagania co do konstrukcji

Wszystkie funkcje oraz przeciążenia operatorów muszą być opisane. Wspomniany opis musi zawierać:

1. ogólną informację co dana funkcja robi,
2. warunki wstępne (o ile są konieczne),
3. opis parametrów wywołania funkcji,
4. opis tego co dana funkcja zwraca.

W trakcie kompilacji finalnej wersji programu nie mogą generować się żadne ostrzeżenia. Kompilację należy wykonywać z opcjami `-Wall`**,** `-pedantic` oraz `-std=c++11`

5 Przygotowanie do zajęć

Przed zajęciami należy napisać kod definicji operacji dodawania symboli. Wspomniana definicja musi być zgodna z dalszym opisem zadania. Należy to zrobić analogicznie do postaci przedstawionej na wykładzie. Operację należy zdefiniować w postaci zwykłej funkcji:

```
Symbol Dodaj(Symbol Arg1, Symbol Arg2);
```

Jak też w postaci funkcji operatorowej jako przeciążenia operatora `'+'`.

```
Symbol operator + (Symbol Arg1, Symbol Arg2);
```

Definicje obu tych funkcji należy wpisać do dostarczonego w ramach materiałów pomocniczych plik `dzialanie.cpp`. Poprawność zapisu należy zweryfikować poprzez skompilowanie pliku.

6 Uproszczenia

Jeżeli zakres zadania okaże się zbyt duży, można ograniczyć się do implementacji tylko operacji dodawania i mnożenia (zarówno jeśli chodzi o funkcje, jak też przeciążenia operatorów). Powoduje to jednak otrzymanie niższej oceny.