Elementy języka C++: klasy, konstruktor, destruktor, funkcje zaprzyjaźnione.

1. **Dwuwymiarowy wektor**
   Napisz klasę Vector reprezentującą dwuwymiarowy wektor. Zaimplementuj: 1) konstruktor bezargumentowy inicjalizujący współrzędne kartezjańskie wektora zerami oraz konstruktor dwuargumentowy inicjalizujący je swoimi argumentami; 2) funkcję składową length zwracającą długość wektora; 3) jednoargumentowy operator - oznaczający wektor przeciwny do danego; 4) operatory dodawania i odejmowania dwóch wektorów oraz operatory += i -=; 5) operatory lewo- i prawostronnego mnożenia wektora przez liczbę oraz operator *=; 6) operator * iloczynu skalarnego dwóch wektorów; 7) operator « wypisywania współrzędnych wektora do strumienia typu ostream oraz operator » wczytywania współrzędnych ze strumienia istream. Współrzędne wektora zapisujemy jako dwie liczby oddzielone spacją.

2. **Kwestionariusz osobowy**
   Napisz klasę Name, której obiekt pamięta imię i nazwisko. Zaimplementuj: 1) konstruktor bezargumentowy inicjalizujący imię i nazwisko łańcuchami pustymi; 2) dwuargumentowy konstruktor inicjalizujący imię i nazwisko swoimi argumentami; 3) funkcję składową initials zwracającą łańcuch zawierający inicjały. Przyjmij dla uproszczenia, że imiona i nazwiska są jednoczłonowe; 3) operator < wyznaczający kolejność alfabetyczną. Operator powinien porównywać najpierw nazwiska, a jeżeli są takie same, to imiona; 4) operator « wypisujący imię i nazwisko do strumienia typu ostream oraz operator » wczytujący je ze strumienia istream.

3. **Łączenie rezystancji**
   Napisz klasę Resistor reprezentującą opornik o zadanym oporze. Zaimplementuj: 1) funkcję składową set ustawiającą opór opornika. Deklaracja: void Resistor::set (double resistance); 2) funkcję zaprzyjaźnioną resistance zwracającą opór opornika. Deklaracja: double resistance (const Resistor &resistor); 3) funkcje zaprzyjaźnione serial i parallel tworzące oporniki odpowiadające szeregowemu i równoległemu połączeniu dwóch oporników. Deklaracja: Resistor serial (const Resistor &first, const Resistor &second); Resistor parallel (const Resistor &first, const Resistor &second). Napisz program, który wczytuje ze standardowego wejścia dwa opory, a następnie wypisuje na standardowe wyjście opór powstały z ich szeregowego oraz równoległego połączenia.

4. **Odwrócony Pan Tadeusz 3 ***
   Napisz klasę Stack reprezentującą stos łańcuchów tekstowych. Zaimplementuj: 1) konstruktor bezargumentowy; 2) funkcję składową push odkładającą element na stos; 3) funkcję składową pop zdejmującą element ze stosu i zwracającą informację o tym, czy operacja się powiodła, to znaczy czy stos nie był pusty; 4) destruktor zwalniający całą pamięć używaną przez stos. Napisz program, który wczytuje ciąg słów, a następnie wypisuje je w odwrotnej kolejności, oddzielone spacjami. Program przetestuj na tekście *Pana Tadeusza* i *Hamleta*.

5. **Liczby wymierne jako ułamki**
   Napisz klasę Rat reprezentującą liczby wymierne p/q. Liczby p i q powinny być pamiętane jako względnie pierwsze z q dodatnim. Zaimplementuj: 1) konstruktor, który można wywoływać bez argumentów lub z jednym albo dwoma argumentami całkowitymi. W pierwszym przypadku tworzony obiekt powinien być inicjalizowany wartością zero, w drugim liczbą całkowitą, a w trzecim ilorazem dwóch argumentów; 2) funkcje składowe numerator i denominator zwracające odpowiednio licznik i mianownik liczby; 3) operator konwersji do typu double; 4) jednoargumentowy operator - oznaczający liczbę przeciwną; 5) operator <; 6) operator preinkrementacji; 7) operatory += i -=; 8) operatory dodawania i mnożenia; 9) operator « wypisujący reprezentowaną liczbę wymierną do strumienia typu ostream oraz operator » wczytujący tę liczbę ze strumienia typu istream. Konstruktor i operator » powinny działać poprawnie również wtedy, gdy podane p i q nie są względnie pierwsze lub q jest ujemne.

6. **Relatywistyczne prawo składania prędkości**
   Napisz klasę Velocity reprezentującą prędkość relatywistyczną w ruchu jednowymiarowym. Zaimplementuj: 1) konstruktor, który można wywołać bez argumentów lub z jednym argumentem rzeczywistym. W pierwszym przypadku tworzony obiekt powinien być inicjalizowany wartością zero, a w drugim wartością podaną jako argument; 2) metodę gamma zwracającą wartość czynnika Lorentza; 3) operator += zgodny z relatywistycznym prawem składania prędkości; 4) operator dodawania zgodny z relatywistycznym prawem składania prędkości; 5) operator « wypisujący prędkość do strumienia typu ostream oraz operator » wczytujący prędkość ze strumienia typu istream.

7. **Wzór Herona na pole trójkąta**
   Napisz klasę Triangle reprezentującą trójkąt o zadanych długościach boków. Zaimplementuj: 1) konstruktor trójargumentowy inicjalizujący długości boków swoimi argumentami; 2) funkcję składową scale przekształcającą trójkąt przez podobieństwo w skali zadanej swoim argumentem; 3) funkcję zaprzyjaźnioną areazwracającą pole trójkąta. Napisz program, który wczytuje długości boków trójkąta, wypisuje pole, a następnie wczytuje liczbę rzeczywistą i wypisuje pole wyjściowego trójkąta przekształconego przez podobieństwo w skali zadanej tą liczbą. *Wskazówka*: Pole trójkąta S wyraża się przez długości boków a, b, c wzorem Herona: $S = \sqrt{p(p-a)(p-b)(p-c)}$, gdzie p jest połową obwodu.

Pytania, a także rozwiązania zadań, można wysyłać na adres: MDABROWSKI@FUW.EDU.PL.