KM PROGRAMS

- **PODSTAWY PROGRAMOWANIA**

Zad 1

Do tablicy jednowymiarowej każdy kolejny element losuj z przedziału <4,10> dopóki suma aktualnie losowanej wartości i wszystkich poprzednich elementów tablicy nie będzie liczbą pierwszą. Wypisz tablicę na ekranie.

Zad 2

Pobieraj od użytkownika trzy liczby, dopóki suma cyfr dwóch pierwszych liczb nie będzie liczbą pierwszą większą od liczby trzeciej.

Zad 3

Elementy pobranej od użytkownika tablicy typu double zwiększ o średnią elementów z tablicy znajdujących się pod indeksami parzystymi.

Zad 4

Do tablicy jednowymiarowej liczb całkowitych wylosuj elementy z przedziału <5, 20>. Następnie zaimplementuj nową tablicę, w której każdy kolejny element na pozycji i począwszy od i = 1 jest elementem, którego wartość to suma cyfr elementów pierwszej tablicy z pozycji od 0 do i-1. Pierwszy element nowej tablicy ma wartość 0. Wypisz w konsoli dwie otrzymane tablice.

Zad 5

Pobieraj od użytkownika współrzędne trzech punktów dopóki nie będą one współliniowe, czyli nie będą leżały na jednej prostej.

- **TABLICE**

Zad 1

Wymiary tablicy dwuwymiarowej losujemy z przedziału <3,10> dopóki nie będzie kwadratowa. Elementy tablicy są losowane z przedziału <4, 20>. Wypisz na ekranie zestawienie numerów tych wierszy i kolumn, które mają sumy elementów różniące się o mniej niż 3.

Zad 2

Wymiary tablicy dwuwymiarowej podaje użytkownik dopóki ilość wierszy nie będzie liczbą parzystą większą od 2. Ilość kolumn jest pobierana dopóki nie będzie liczbą większa od 4. Elementy tablicy są losowane z przedziału <5, 25>. Następnie zaimplementuj drugą tablicę o identycznych wymiarach co pierwsza. Elementy tej tablicy to sumy cyfr elementów występujących na odpowiadających pozycjach w tablicy pierwszej. Następnie dodaj dwie tablice do siebie według zasady pierwszy wiersz pierwszej tablicy z ostatnim wierszem drugiej tablicy, drugi wiersz pierwszej tablicy z przedostatnim wierszem drugiej tablicy itd.

Zad 3

Wypełnij tablicę kwadratową o wymiarach 10 na 10 elementami losowymi z przedziału <10,100>. Następnie podaj liczbę, która będzie indeksem kolumny i wiersza o tym samym numerze, które posiadają największą sumę elementów. Przykład:

```
Tablica:   1 2 8
           4 2 5
           9 8 1
Suma wiersz 0 oraz kolumna 0 -> 25
Suma wiersz 1 oraz kolumna 0 -> 23
Suma wiersz 2 oraz kolumna 2 -> 32
Wynik -> 2
```

Zad 4

Tablica kwadratowa dwuwymiarowa o wymiarach 10 na 10. Elementy tablicy losujemy z przedziału <45, 230>. Oblicz ile elementów na przekątnej tablicy ma wartość większą od wszystkich elementów w tym samym wierszu co sprawdzany element na przekątnej. Przykład:

3 1 2 -> element na przekątnej jest największy w swoim wierszu

8 2 2 -> element na przekątnej NIE jest największy w swoim wierszu

6 2 9 -> element na przekątnej jest największy w swoim wierszu

Wynik -> 2 elementy

Zad 5

Tablica nieregularna. Ilość wierszy podawana jest przez użytkownika. Dla każdego wiersza ilość kolumn jest losowana z przedziału <3, 15>. Elementy tablicy są losowane z przedziału <4, 20>. Oblicz ile wierszy posiada pierwszy element większy od średniej arytmetycznej pozostałych elementów. Wiersze o tej własności wyzeruj. Wypisz w konsoli zmodyfikowaną tablicę.

- **NAPISY**

Zad 1

Stosując wyrażenia regularne pobieraj od użytkownika napis dopóki nie będzie składał się z trzech wyrazów o długościach dokładnie 10 znaków. Pierwszy wyraz ma zawierać same duże litery. Drugi wyraz ma zawierać same małe litery, trzeci wyraz ma zawierać same cyfry. Napis wynikowy ma powstać poprzez zamianę znaków napisu trzeciego na odpowiednich pozycjach z literami pierwszego napisu, kiedy rozpatrujemy znaki na pozycjach parzystych oraz poprzez zamianę znaków napisu trzeciego na odpowiednich pozycjach z literami napisu drugiego, kiedy rozpatrujemy znaki na pozycjach nieparzystych. Przykład:

Dane: ABCDEFGHIJ abcdefghij 0123456789.

Wynik: 0B2D4F6H8J a1c3e5g7i9 AbCdEfGhIj

Zad 2

Pobieraj od użytkownika napis, dopóki jego długość nie będzie kwadratem liczby mniejszej od 10. Następnie załaduj go do kwadratowej tablicy dwuwymiarowej wpisując do niej kolejne znaki wierszami. Napis wynikowy otrzymujemy poprzez odczytanie elementów tablicy dwuwymiarowej kolumnami i sklejenie ich w jeden napis.

Zad 3

Pobierz od użytkownika 10 adresów. Stosując wyrażenia regularne dopilnuj, żeby każdy adres miał postać:

ul. NazwaUlicy NumerUlicy KodPocztowy Miasto

NazwaUlicy – wyraz składający się z samych dużych liter

Numer ulicy – liczba z przedziału 10 – 99

KodPocztowy – standardowy kod w formacie xx-xxx

Miasto – jedno z Kraków, Warszawa, Wrocław

Następnie sprawdź, w którym mieście znajduje się najwięcej adresów spośród pobranych.

Zad 4

Pobierz od użytkownika 10 napisów w jednym wierszu. Napisy mają być oddzielone znakiem spacji oraz składać się z pierwszej i ostatniej dużej litery alfabetu. Na pozostałych pozycjach w napisach muszą występować tylko i wyłącznie cyfry. Zastosuj wyrażenia regularne. Pobrane napisy zmodyfikuj w ten sposób, że występujące w środku napisu cyfry zastąp ich sumą, przykładowo: A162B zamień na A9B. po zmodyfikowaniu wszystkich napisów posortuj je malejąco, sklej w jeden napis i wypisz w konsoli.

Zad 5

Spośród 5 pobranych wcześniej od użytkownika napisów losuj trzy napisy tak długo, aż wszystkie wylosowane nie będą swoimi anagramami. Jeżeli próba losowań przekroczy 100 wypisz komunikat BŁĄD LOSOWANIA.

- **JAVA 8 - STRUMIENIE**

Zad 1

Pobierz od użytkownika listę dowolnych marek samochodów. Następnie stosując strumienie zwróć kolekcję Set marek, które mają w nazwie składającej się z samych dużych liter co najmniej 3 samogłoski, których suma kodów ASCII jest liczbą parzystą o nieparzystej cyfrze dziesiątek.

Zad 2

Pobierz od użytkownika listę dowolnych liczb całkowitych. Następnie stosując strumienie odrzuć liczby, których kwadraty są liczbą większą od 100 i posortuj pozostałe liczby według kryterium: największa liczba to ta, która posiada największą cyfrę dziesiątek. Posortowane elementy wypisz w konsoli.

Zad 3

Pobierz od użytkownika listę dowolnych dat. Następnie stosując strumienie odrzuć te daty, które mają więcej niż 20 lat. Następnie przerób strumień dat na strumień liczb całkowitych, które odpowiadają numerowi miesiąca daty. Otrzymany strumień posortuj malejąco i zwróć w postaci kolekcji Set.

Zad 4

Sporządź listę obiektów klasy Samochód. Każdy samochód ma pole marka, cena, pojemność. Klasę należy zaopatrzyć w niezbędne metody potrzebne do jej prawidłowego działania i zarządzania nią. Stosując strumienie zwróć średnią arytmetyczną cen samochodów w strumieniu oraz w stosując drugi strumień posortowaną listę samochodów malejąco według pojemności.

- **OBIEKTOWOŚĆ I KOLEKCJE**

Zad 1

Klasa Klient zawiera pola składowe imię, nazwisko, numer konta, saldo. Dla wymienionych pól przygotuj akcesory. Imię oraz nazwisko mają składać się z samych dużych liter. Numer konta dla uproszczenia ma być reprezentowany przez napis zawierający dokładnie 12 cyfr, których suma musi być parzysta, natomiast saldo musi być nieujemne. Zaimplementuj dla klasy konstruktory. Konstruktor argumentowy przyjmuje argumenty, które następnie są przypisywane do pól składowych, natomiast konstruktor bezargumentowy pobiera dane od użytkownika. Oprócz tego klasa powinna posiadać wszelkie metody potrzebne do prawidłowego działania klasy. Programiści C++ do wypisywania danych klienta muszą zastosować przeładowany operator <<. Klasa zawiera metodę, której zadaniem jest symulacja obsługi klienta. W przypadku klasy Klient procedura obsługi klienta polega na zapytaniu o numer konta. Jeżeli klient trzy razy poda zły numer konta, czyli taki, który nie jest do niego przypisany, metoda kończy się komunikatem BLAD NUMERU KONTA. W przeciwnym razie wyświetlana jest informacja o stanie konta (wartość zmiennej saldo) i prośba o wybranie jednej z dwóch opcji: wpłata lub wypłata gotówki. Po wybraniu klient podaje kwotę i w zależności od rodzaju operacji stan konta jest zmniejszany lub zwiększany o podaną kwotę. Klient nie może wypłacić więcej niż pozwala mu na to saldo. Stan salda nie może przekroczyć 1000000.

Klasa KlientBiznesowy dziedziczy po klasie Klient i posiada dodatkowo pole podatek, które określa procent podatku, który pobierany jest za każdą transakcję, dodatkowo zmniejszając stan salda. Podatek obliczany jest od wpłacanej lub wypłacanej kwoty. Wysokość podatku musi zawsze zawierać się w przedziale <10,20> procent. Dodatkowo klasa zawiera zmienną określającą ilość transakcji. Jeżeli wartość tej zmiennej przekroczy wartość ustaloną za pomocą stałej, która również jest w tej klasie, klient płaci 2% podatku od aktualnego stanu salda. Na podstawie

powyższych informacji przygotuj akcesory oraz konstruktory dla klasy. Konstruktor argumentowy posiada argumenty przypisywane do pól składowych klasy, natomiast konstruktor bezargumentowy pobiera wszystkie dane od użytkownika. Oprócz tego klasa zawiera wszystkie metody niezbędne do prawidłowego funkcjonowania klasy.

Klasa Bank zawiera kolekcję uporządkowaną klientów. Należy w konstruktorze argumentowym pobrać przykładowych klientów z pliku tekstowego lub bazy danych (do ustalenia) . Argumentem konstruktora jest nazwa pliku tekstowego lub parametry połączenia z bazą danych. Klasa zawiera metodę, które ma za zadanie wykonać symulację obsługi każdego klienta z listy, a na koniec wypisać imię i nazwisko każdego klienta oraz stan jego konta.

Przetestuj opisane wyżej klasy w metodzie głównej programu.

Zad 2

Interfejs Figura posiada bezargumentową metodę pole, która zwraca wartość typu double. Interfejs FiguraSymetriaSkalowanie implementuje interfejs Figura i posiada metody:

void skaluj(double skala),

void symetriaSrodek(),

void symetriaX(),

void symetriaY(),

double odleglosc().

Klasa Prostokąt przechowuje w polach prywatnych współrzędne lewego górnego wierzchołka oraz długości boków. Klasa zawiera wszystkie niezbędne do działania metody oraz implementuje interfejs FiguraSymetriaSkalowanie. Metoda skaluj w przypadku prostokąta polega na przeliczaniu długości boków przez podaną jako argument wartość

skali. Metoda symetriaSrodek wykonuje standardową procedurę przekształcenia prostokąta w symetrii środkowej względem początku układu współrzędnych, natomiast metody symetriaX oraz symetriaY przekształcają prostokąt w symetrii osiowej względem odpowiednio osi x oraz osi y. Metoda odległość zwraca odległość punktu przecięcia przekątnych prostokąta od początku układu współrzędnych. Metoda pole zwraca pole prostokąta.

Klasa Trójkąt posiada sześć zmiennych składowych typu double, które przechowują współrzędne wierzchołków trójkąta. Klasa implementuje interfejs FiguraSymetriaSkalowanie implementując wszystkie metody według tych samych reguł matematycznych, które opisano dla klasy Prostokąt. Metoda odległość zwraca odległość środka ciężkości trójkąta od początku układu współrzędnych.

Klasa FiguraKolekcja zawiera pole składowe – kolekcję, która przechowuje instancje interfejsu Figura. Dodatkowo klasa posiada metody:

dodajFigura(Figura f) – dodaje do kolekcji figurę podaną jako argument

zwrocFiguraMinPole() – zwraca instancję interfejsu Figura o najmniejszym polu

zwrocFiguraMaxOdleglosc() – zwraca instancję interfejsu Figura o największej odległości od początku układu współrzędnych.

Klasa zawiera również metodę statyczną testerFiguraKolekcja, która przyjmuje instancję klasy FiguraKolekcja oraz wypisuje w konsoli wszystkie elementy kolekcji znajdujące się w kolekcji oraz figury o najmniejszym polu oraz największej odległości od początku układu współrzędnych.

Przetestuj zaimplementowaną hierarchię klas oraz interfejsów w metodzie głównej programu.

Zad 3

Klasa Słownik reprezentuje mały słownik, który zawiera tłumaczenia słowa zapisanego w języku Polskim na inne języki. Klasa posiada pole składowe - mapę, w której klucz to napis, a wartość reprezentowana jest przez listę elementów typu String. Przygotuj konstruktor bezargumentowy inicjalizujący pustą mapę oraz metodę wypisującą zawartość mapy. Klasa zawiera dwie metody abstrakcyjne:

void pobierzTlumaczenie()

void zapiszTlumaczenie()

Klasa SłownikPlik rozszerza klasę Słownik i implementuje metody abstrakcyjne w następujący sposób:

void pobierzTlumaczenie() – polskie słowo oraz odpowiadające mu tłumaczenia pobierane są z pliku tekstowego jezyki.txt, w którym każdy wiersz ma postać:

slowo_pl EN_slowo_angielskie GER_slowo_niemiecie FR_slowo_francuskie

Pobrane dane należy umieścić w mapie, gdzie kluczem będzie słowo polskie, natomiast pozostałe słowa już bez przedrostków EN_, GER_, FR_ zostaną zapisane w liście w kolejności slowo_angielskie, slowo_niemieckie, slowo_francuskie. Uwaga w pliku jezyki.txt słowa angielskie, niemieckie, francuskie nie muszą zawsze znajdować się w tej samej kolejności.

void zapiszTlumaczenie() - metoda zapisuje do pliku o nazwie noweTlumaczenie.txt wszystkie tłumaczenia w zachowanej w mapie, przekonwertowane na napisy składające się z samych dużych liter.

Klasa SlownikKonsola rozszerza klasę Słownik i implementuje metody abstrakcyjne w następujący sposób:

void pobierzTlumaczenie() – użytkownik podaje polskie słowo, później skrót EN, GER lub FR w celu ustalenia jakie tłumaczenie podaje. Należy dopilnować, żeby tłumaczenie dla każdego języka podano dokładnie jeden

raz, następnie umieścić w mapie polskie słowo, a później tłumaczenia w kolejności angielskie, niemieckie, francuskie. Jeżeli jest tylko taka możliwość należy stosować wyrażenia regularne.

void zapiszTlumaczenie() - metoda pobiera od użytkownika informacje, które tłumaczenia mają zostać wypisane w konsoli używając skrótów EN, GER, FR. Należy dopilnować, żeby każde tłumaczenie pojawiło się dokładnie jeden raz. Później metoda wypisuje na ekranie wybrane tłumaczenia zachowując kolejność angielskie, niemieckie, francuskie

W głównej metodzie programu przetestuj działanie zaimplementowanych klas.