1. Zaimplementować klasę `Pomiary`, która przechowuje wykonywane pomiary (liczby całkowite) i umożliwia odczyt pewnych wartości statystycznych.
Metody publiczne:
•`Pomiary();`
Tworzy nowy pusty obiekt (nie zawierający żadnych pomiarów.
•`void dodaj(int wartosc);`
Rejestruje nowy pomiar o podanej wartości.
•`int ilePomiarow();`
Zwraca liczbę zarejestrowanych pomiarów.
•`double srednia();`
Zwraca średnią wartość wszystkich zarejestrowanych pomiarów.
•`int najwiekszy();`
Zwraca wartość największego zarejestrowanego pomiaru.

2. Zaimplementować klasę `Napis`, której obiekty reprezentują napisy.
Metody publiczne:
•`Napis();`
Tworzy nowy pusty napis.
•`Napis(char* s);`
Tworzy nowy napis o podanej treści.
•`void dolacz(char znak);`
Dołącza na koniec napisu podany znak.
•`char znak(int pozycja);`
Zwraca znak na podanej pozycji.
•`void wypisz();`
Wypisuje napis na strumień `cout`.

3. Zaimplementować klasę `Trojkat`, której obiekty reprezentują trójkąty na płaszczyźnie.
Zakładamy, że zdefiniowana została następująca struktura reprezentująca punkty:

```
struct Punkt {
     double x;
     double y;
};
```

Metody publiczne:
•`Trojkat(Punkt p1, Punkt p2, Punkt p3);`
Tworzy nowy trójkąt o podanych wierzchołkach.
•`void przesun(double dx, double dy);`
Przesuwa trójkąt o zadany wektor.
•`bool nalezy(Punkt p);`
Zwraca true jeśli dany punkt należy do trójkąta.

Następnie zmodyfikować tą klasę tak, aby trójkąty można było porównywać przy użyciu operatorów == i `!=`.

4. Zaimplementować klasę `Osoba`, której obiekty przechowują informacje o osobach.

Metody publiczne:
•`Osoba(char* imie, char* nazwisko);`
Tworzy osobę o zadanym imieniu i nazwisku.
•`void ustawWiek(int wiek);`
Ustawia wiek osoby.
•`bool starszaNiz(const Osoba& osoba);`
Zwraca true, jeśli osoba jest starsza niż podana jako argument.
• `bool imiennik(const Osoba& osoba);`
Zwraca true, jeśli imiona osób sa takie same.

5. Zaimplementować klasę `TablicaPosortowana`, której obiekty przechowują kolekcję liczb całkowitych posortowanych rosnąco.
Metody publiczne:
•`TablicaPosortowana(int maksymalnyRozmiar);`
Tworzy nową pustą tablicę, która może przechowywać nie więcej niż `maksymalnyRozmiar` elementów.
•`int dodaj(int wartosc);`
Dodaje nowy element.
•`bool usun(int wartosc);`
Jeśli podany element znajduje się w tablicy, to go usuwa i zwraca true; jeśli nie zwraca false.
•`int element(int indeks);`
Zwraca element o podanym indeksie, np. element(0) to najmniejszy element, element(1) drugi najmniejszy, itd.
•`int max();`
Zwraca największy element.

6. Zaimplementować klasę `Konto` o następujących metodach:
Metody publiczne:
•`Konto();`
Tworzy nowe konto o stanie 0.
•`int stanKonta();`
Zwraca stan konta.
•`void zmien(int oIle);`
Zmienia stan konta o podaną wartość.
•`int sumaUznan();`
Zwraca sumę uznań (dodatnich zmian stanu).
•`int sumaObciazen();`
Zwraca sumę obciążeń.

Następnie zmodyfikować klasę tak, aby móc ustanowić maksymalny dopuszczalny debet i żeby operacje prowadzące do przekroczenia tego debetu nie były wykonywane.

7. Zaimplementować klasę `Histogram`, która zapamiętuje wyniki pomiarów w podziale na zadane przedziały.
Metody publiczne:
•`Histogram(double min, double max, int ilePrzedzialow);`
Tworzy nowy obiekt, który obsługuje wartości z przedziału [min,max] podzielone na `ilePrzedzialow` równych części.
•`int dodaj(double wartosc);`

Dodaje nową wartość.
•`int ileWPrzedziale(int numer);`
Zwraca liczbę wartości należącą do podanego przedziału (numeracja zaczyna się od 1).
•`int ilePozaZakresem();`
Zwraca liczbę wartości poza obsługiwanym zakresem.

Następnie zmodyfikować tą klasę tak, aby do liczby pomiarów w danym przedziale odwoływać się przez `[]`.

8. Zaimplementować klasę `Set` zliczającą punkty w secie. Gra toczy się między graczami A i B do określonej liczby punktów, ale zwycięzca musi mieć 2 punkty przewagi, aby set został zakończony. Uwaga: jeśli set został ukończony zaliczanie punktów nie powinno być możliwe.
Metody publiczne:
•`Set(int doIlu);`
Tworzy nowy set grany do `doIlu` punktów (i do 2-punktowej przewagi).
•`void punktA();`
Zalicza punkt graczowi A.
•`void punktB();`
Zalicza punkt graczowi B.
•`void drukujWynik();`
Drukuje wynik, np. tak: `22:17`
•`char zwyciezca();`
Jeśli set został ukończony to zwraca zwyciężcę ('A' lub 'B'); jeśli nie zwraca '?'.
•`bool koniec();`
Zwraca true jeśli set został ukończony.

9. Zaimplementować klasę Czas, której obiekty reprezentują godziny i minuty.
Metody publiczne:
`class Czas {`
•`Czas();`
Tworzy obiekt ustawiony na północ (0:00).
•`Czas(int godzina);`
Równa godzina.
•`Czas(int godzina, int minut);`
Tworzy czas o zadanej wartości.
•`void drukuj24();`
Drukuje czas na cout w formacie 24-godzinnym.
•`void drukuj12();`
Drukuje czas na cout w formacie 12-godzinnym.
•`int roznica(Czas c);`
Zwraca liczbę minut, po jakiej nastąpi godzina c.

Następnie zmodyfikować tą klasę tak, aby operator += dodawał do obiektu odpowiednią liczbę minut.

10. Zaimplementować klasę `Trapez`, której obiekty reprezentują trapezy równoramienne. Istotne będą tylko wymiary trapezów, a nie ich położenie na płaszczyźnie.
Metody publiczne:

- `Trapez(double a);`

tworzy kwadrat o podanym boku,

- `Trapez(double a, double b);`

tworzy prostokąt o podanych bokach,

- `Trapez(double a, double h, double kat);`

tworzy trapez o podstawie `a`, wysokości `h` i kącie przy podstawie `kat` (wyrażonym w stopniach),

- `double pole();`

zwraca pole trapezu,

- `double obwod();`

zwraca obwód,

- `bool rowne(Trapez& t);`

zwraca `true` jeśli trapezy są przystające.
Mamy do dyspozycji funkcje matematyczne (sin, cos, sqrt, itd.) oraz stałą PI.

11. Zaimplementować klasę `Obraz`, której obiekty reprezentują czarno-białe, prostokątne obrazy o ustalonych wymiarach x na y. Obrazy składają się z kwadratowych pikseli, których współrzędne należą do zakresu (0...x-1), (0...y-1), a jasność jest dana przez liczbę całkowitą z zakresu od 0 (czarny) do 100 (biały).
Metody publiczne:

- `Obraz(double sz, double wys);`

tworzy nowy, biały obraz o wymiarach `sz` na `wys`.

- `Obraz(double sz, double wys, int jasnosc);`

tworzy nowy obraz o wymiarach `sz` na `wys`. Trzeci parametr opisuje jasność wszystkich punktów

- `int jasnosc(int x, int y);`

zwraca jasność podanego piksela.

- `void lustro();`

odbija obraz względem osi pionowej (OY).

- `void wklej(Obraz& o, int x, int y);`

wkleja podany obraz tak, aby jego lewy górny róg (tzn. piksel o współrzędnych (0,0)) znalazł się w punkcie (x,y)

- `Obraz wytnij(int x, int y, int sz, int wys);`

zwraca obraz powstały z wycięcia obszaru o lewym górnym rogu (x,y), szerokości `sz` i wysokości `wys`. Jeśli wycinany obszar nie mieści się w obrazie, należy zwrócić mniejszy.

12. Zaimplementować klasę `Znajomi`, której obiekty reprezentują zbiór osób (ponumerowanych 1..n). Każda para osób może się znać lub nie (jest to relacja symetryczna, tzn. jeśli a zna b, to b również zna a).

Metody publiczne:

- `Znajomi(int n);`

tworzy zbiór n osób, żadne dwie się nie znają

- `bool zna(int a, int b);`

zwraca true wtedy i tylko wtedy, gdy a zna b,

- `void poznaj(int a, int b);`

osoby a i b stają się znajomymi,

- `void wspolniZnajomi(int a, int b);`

wypisuje na ekran listę wspólnych znajomych osób a i b,

- `void spotkanie(int a);`

wszyscy znajomi osoby a poznają się ze sobą,

- `int max();`

zwraca osobę która ma najwięcej znajomych (jeśli takich jest wiele, którąkolwiek z nich).

13. Dany jest typ

```
class Karta { public: int kolor; int ranga; };
```

reprezentujący karty do gry. Pole kolor może przyjmować wartości 1..4, a pole ranga wartości 1..13. Zaimplementować klasę `ZbiorKart`, której obiekty reprezentują zbiory kart z 52-elementowej talii.

Metody publiczne:

- `ZbiorKart();`

tworzy pusty zbiór kart.

- `void dodaj(Karta k);`

dodaje nową kartę do zbioru.

- `Karta losowa();`

zwraca losową kartę należącą do zbioru. Prawdopodobieństwo wylosowania każdej karty powinno być jednakowe. Można użyć funkcji

`int losuj(int n);`

która zwraca losową liczbę z zakresu 1..n.

- `int ileWKolorze(int kolor);`

zwraca liczbę kart w podanym kolorze.

- `bool sekwens();`

zwraca true wtedy i tylko wtedy, gdy zbiór zawiera trzy karty w tym samym kolorze o sąsiadujących rangach (np. 4,5,6).

14. Zaimplementować klasę `Kolo`, której obiekty reprezentują koła na płaszczyźnie.

Metody publiczne:

- `Kolo(double x, double y, double r);`

tworzy koło o środku w punkcie (`x`,`y`) i promieniu `r`,

- `void przesun(double dx, double dy);`

przesuwa koło o wektor (`dx`, `dy`),

- `bool zawiera(double x, double y);`

zwraca `true` wtedy i tylko wtedy, gdy koło zawiera punkt (`x`,`y`),

- `bool zawiera(Kolo k);`

zwraca `true` wtedy i tylko wtedy gdy dane koło zawiera koło `k`.

- `Kolo przechodzace(double x, double y);`

zwraca koło współśrodkowe przechodzące przez punkt (`x`,`y`);

Uwaga: funkcja `sqrt` zwraca pierwiastek parametru.

15. Zaimplementować klasę `Prostokat`, której obiekty reprezentują prostokąty na płaszczyźnie. Zakładamy, że boki prostokątów są równoległe do osi układu współrzędnych.

Metody publiczne:

- `Prostokat(double x1, double y1, double x2, double y2);`

tworzy prostokąt o wierzchołkach (`x1`,`y1`) i `(x2,y2)`,

- `Prostokat(double x, double y, double a);`

tworzy kwadrat o środku (`x`, `y`) i długości boku a,

- `double pole();`

zwraca pole prostokąta,

- `Prostokat czescWspolna(Prostokat p);`

zwraca część wspólną z podanym prostokątem (można założyć że jest niepusta).

- `void powieksz(double c);`

powiększa rozmiary prostokąta `c` razy. Środek prostokąta powinien zostać zachowany.

16. Zaimplementować klasę `Macierz`, której obiekty reprezentują kwadratowe macierze liczb rzeczywistych (tj. typu double).

Metody publiczne:

- `Macierz(int rozmiar);`

tworzy zerową macierz o podanym rozmiarze.

- `Macierz(int rozmiar, double x);`

tworzy macierz o podanym rozmiarze. Wartości na przekątnej powinny być równe x, poza nią 0.

- `Macierz suma(Macierz& m);`

zwraca sumę macierzy. Można założyć, że parametr m ma taki sam rozmiar jak dana macierz.

- `double slad();`

zwraca sumę elementów na przekątnej.

- `void trans();`

transponuje macierz, tzn zamienia elementy o współrzędnych (a,b) z elementami o współrzędnych (b,a).

- `double min();`

zwraca najmniejszy element macierzy.

17. Zaimplementować klasę `Trojmian`, której obiekty reprezentują trójmiany kwadratowe postaci ax^2+bx+c. Uwaga: funkcja sqrt zwraca pierwiastek kwadratowy z podanej liczby.

Metody publiczne:

- `Trojmian(double a, double b, double c);`

tworzy trójmian o podanych współczynnikach.

- `Trojmian(double x1, double x2);`

tworzy trójmian, którego współczynnik a jest równy 1, a pierwiastki to x1 i x2,

- `int ilePierwiastkow();`

zwraca liczbę pierwiastków rzeczywistych trójmianu,

- `void pomnoz(double s);`

mnoży trójmian przez podaną liczbę.

- `void drukuj();`

drukuje współczynniki a,b,c na ekranie.

18. Zaimplementować klasę `Tablica`, której obiekty reprezentują nieskończone tablice liczb typu int. Każda tablica zawiera elementy o indeksach 0,1,2,3,.. itd.

Metody publiczne:

- `Tablica();`

tworzy nową tablicę wypełnioną zerami.

- `void ustaw(int indeks, int wartosc);`

ustawia podany element o podanym indeksie.

- `int wartosc(int indeks);`

zwraca element o podanym indeksie.

- `int suma();`

zwraca sumę elementów tablicy.

- `void ustaw(int a, int b, int wartosc);`

ustawia elementy o indeksach a, a+1, …, b-1 na zadaną wartość.

- `void przesun();`

przesuwa elementy tablicy o 1 w lewo, tzn. element indeksie 0 staje się równy elementowi o indeksie 1, element o indeksie 1 elementowi o indeksie 2, itd.