ZAAWANSOWANE PROGRAMOWANIE KOMPUTEROWE WNE (2025)

2. Klasy

(1) Zaimplementować klasę

```
class Punkt {
public:
   Punkt();
   Punkt(double _x, double _y);
   void drukuj() const;
   void przesun(double dx, double dy);
   double odleglosc(const Punkt& p) const;
   double x;
   double y;
};
```

Opis metod:

- `Punkt()`: tworzy punkt (0,0).
- `Punkt(double x, double y)`: tworzy punkt (`x`,`y`).
- `void drukuj()`: drukuje współrzędne na ekranie.
- `void przesun(double dx, double dy)`: przesuwa punkt o zadany wektor.
- `double odleglosc(const Punkt& p)`: zwraca odległość do podanego punktu.

(2) Zaimplementować klasę `Czas`. Obiekty tej klasy reprezentują czas w ciągu dnia z dokładniością do minuty.

```
class Czas {
public:
   Czas();
   Czas(int godzina, int minuta);
   void drukuj24() const;
   void drukuj12() const;
   int dodajMinuty(int ileMinut);
};
```

Opis metod:

- `Czas()`: tworzy północ (godzinę 0:00).
- `Czas(int godzina, int minuta)`: tworzy podaną godzinę.
- `void drukuj24()`: drukuje na ekranie w formacie 24-godzinnym.
- `void drukuj12()`: drukuje na ekranie w formacie 12-godzinnym.

- `int dodajMinuty(int ileMinut)`: dodaje podaną liczbę minut. Liczba minut może być ujemna.

(3) Zaimplementować klasę `Parking`. Obiekty tej klasy pozwalają zapamiętać, które miejsca na parkingu są wolne, a które zajęte. Miejsca są ponumerowane od 1 do n.

```
class Parking {
public:
   Parking(int ileMiejsc);
   bool zajmij(int a);
   bool zwolnij(int a);
   int ileWolnych();
   int znajdzWolne();
};
```

Opis metod:

- `Parking(int ileMiejsc)`: tworzy parking z miejscami 1,. . . ,ileMiejsc.
- `bool zajmij(int a)`: jeśli miejsce o numerze `a` jest wolne i zwraca `true`. Jeśli to miejsce jest już zajęte albo `a` nie jest poprawnym numerem miejsca, nic się nie dzieje i zwraca `false`.
- `bool zwolnij(int a)`: zwalnia miejsce o numerze `a`.
- `int ileWolnych()`: zwraca liczbę wolnych miejsc.
- `int znajdzWolne()`: zwraca numer dowolnego wolnego miejsca (lub 0 jeśli wszystkie są zajęte).

(4) Zaimplementować klasę `Kolejka`, której obiekty reprezentują ciągi napisów typu `string`.

```
class Kolejka {
public:
   Kolejka();
   void dodaj(string a);
   string usun();
   int ile() const;
   string operator[](int i);
};
```

Opis metod:

- `Kolejka()`: tworzy pustą kolejkę.
- `void dodaj(string a)`: dodaje napis na końcu kolejki.
- `string usun()`: usuwa i zwraca pierwszy napis.
- `int ile()`: zwraca liczbę elementów kolejki.
- `string operator[](int i)`: zwraca napis o podanym indeksie.

(5) 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 operator `[]`.

(6) Zaimplementować klasę `Macierz`, której obiekty reprezentują macierze o współczynnikach `double`. Zaimplementować operacje umożliwiające obliczenia na macierzach.

(7) Zaimplementować klasę `Wymierna` umożliwiającą rachunki na liczbach wymiernych.