Zadanie 1. Zdefiniuj klasę reprezentującą pracownika. Składowe powinny reprezentować nazwisko pracownika, staż pracy, miesięczne wynagrodzenie netto, dodatki do wynagrodzenia, potrącenia od wynagrodzenia oraz kwotę do wypłaty. Metody powinny natomiast realizować następujące operacje:

- tworzenie obiektu i jego inicjalizację,
- wyświetlanie informacji na temat pracownika,
- aktualizację kwoty dodawanej do wynagrodzenia,
- aktualizację kwoty potrąceń od wynagrodzenia,

Napisz program, w którym wszystkie składowe klasy zostaną przetestowane.
Rozszerzenie. W klasie reprezentującej pracownika, przeciążyć operator wyjścia.

Zadanie 2. Zaprojektuj klasę reprezentującą wynagrodzenie. Składowe powinny reprezentować stawkę godzinową (wspólną dla wszystkich), liczbę przepracowanych godzin w tygodniu, oraz wynagrodzenie za tydzień pracy. W programie stwórz tablicę obiektów reprezentujących klasę wynagrodzenie. Program powinien prosić użytkownika o podanie liczby godzin przepracowanych przez każdą osobę, a następnie wyświetlić całkowite wynagrodzenie każdej osoby. Weryfikacja danych: liczba przepracowanych godzin nie może być większa niż 60.
Rozszerzenie. W klasie reprezentującej wynagrodzenie, przeciążyć operator wyjścia.

Zadanie 3. Zdefiniuj klasę służącą do reprezentowania informacji dotyczących osoby: nazwisko, rok, miesiąc i dzień urodzenia oraz identyfikator numeryczny. Następnie zaimplementuj następujące metody umożliwiające:
1. wczytywanie danych osoby z klawiatury;
2. uzupełniające pola losowymi danymi wg reguł:
- do pola nazwisko wpisz łańcuch "naz" zakończony losową liczbą należącą do przedziału [0,100);
- do pola rok wpisz losową liczbę całkowitą należącą do przedziału [1960, 2000);
- do pola miesiąc wstaw losową liczbę całkowitą należącą do przedziału [1, 13);
- do pola dzień wstaw losową liczbę całkowitą należącą do przedziału [1, 32);
- do pola id wpisz losową liczbę całkowitą należącą do przedziału [1, 100);
3. wypisywanie zawartości pól na ekran.

Następnie napisz program umożliwiający wybór losowego generowania lub wprowadzania danych osoby, a następnie wyświetlający dane na ekranie.
Rozszerzenie. W klasie reprezentującej osobę, przeciążyć operator wyjścia.

Zadanie 4. Korzystając z klasy z poprzedniego zadania utwórz tablicę mogącą zapamiętać dane dla n osób. Następnie opracuj metody/funkcje:

- wpisującą do tablicy obiekt o losowych polach (jak w zadaniu poprzednim);
- wyświetlającą wszystkie wypełnione elementy tablicy na ekran;
- wyświetlającą elementy tablicy o nazwisku przekazanym jako parametr;
- wyświetlającą na ekran osoby, dla których identyfikatory należą do przedziału [a,b];

Następnie napisz program umożliwiający generowanie danych dla n osób oraz wyświetlenie ich danych według powyższych zasad.

Zadanie 5. Symulator rzutów monetą. Napisz klasę reprezentującą monety, która zawiera tylko informację na temat strony monety, tzn. "orzel" "reszka".
Niech klasa posiada:
- domyślny konstruktor wybierający losowo stronę monety
- metodę symulującą rzut monetą,
- metodę/funkcję zwracającą informację o stronie monety.

Napisz program demonstrujący użycie klasy. W programie „rzuć” n razy monetą, wyświetl wynik rzutów, oraz ile razy wypadła reszka.
Rozszerzenie. W klasie reprezentującej monety, przeciążyć operator wyjścia.

Zadanie 6. Gra hazardowa. Wykorzystując definicję klasy z powyższego zadania (symulator rzutów monetami) w programie stwórz trzy obiekty reprezentujące monety 1 zł, 2 zł, 5 zł.
Początkowo saldo gry niech będzie równe zero. W każdej kolejce program powinien rzucać monetami i dodawać do salda ich wartości, jeżeli wypadnie orzeł. Na przykład jeżeli wypadnie orzeł monety 1zł, saldo powinno zostać powiększone o 1 zł. Jeżeli wypadnie reszka, saldo ma nie być powiększane. Gra powinna się kończyć wtedy, gdy saldo będzie równe lub większe niż 20 zł. Saldo równe dokładnie 20 oznacza wygraną, a większe przegraną.

Zadanie 7. Zdefiniuj klasę Stos, która będzie strukturą typu LIFO – element, który został do tej struktury dodany najpóźniej, będzie z niej wyciągnięty najszybciej. Struktura ma służyć do przechowywania liczb rzeczywistych typu double. Stos zaimplementuj w postaci tablicy tworzonej dynamicznie na stercie.
Funkcjonalność stosu ma być bardzo prosta: kładziemy wartość na stos, ściągamy wartość ze stosu, sprawdzamy jaka wartość znajduje się na wierzchu stosu oraz pytamy o liczbę wszystkich elementów na stosie.
Maksymalna pojemność stosu ma być ustalana w konstuktorze. Potrzebna też będzie informacja o liczbie aktualnie znajdujących się na stosie elementów.
Dodatkowo stos ma mieć zaimplementowaną semantykę kopiowania, tzn. konstruktor kopiujący i operator przypisania.
Napisz program demonstrujący działanie klasy.