klasa map

W informatyce często spotykamy się z przypadkiem, kiedy dostęp do obiektów w kontenerze za pomocą indeksowania liczbami naturalnymi nie jest wystarczający. W takich przypadkach koniecznym jest posiadanie kontenera umożliwiającego dostęp do obiektów za pomocą obiektu dowolnego typu – tak zwanego klucza, który jest odwzorowany przez kontener w odpowiadającą mu wartość. W języku C++ taką rolę pełni kontener `map` będący kontenerem asocjacyjnym, czyli takim, w którym oprócz wartości (danych) przechowywane są również unikalne klucze niekoniecznie będące liczbami naturalnymi. Nazwa `map` związana jest z funkcjonalnością kontenera – odwzorowuje (mapuje) klucze na wartości. Można zatem powiedzieć, że mapa jest uogólnieniem tablicy gdyż indeksem nie musi być liczba naturalna, ale dowolny typ danych. Kontenery asocjacyjne nie przechowują obiektów w kolejności ich dodawania lecz używają własnej kolejności. Przykładem użycia mapy może być książka telefoniczna (odwzorowanie: nazwisko (`string`)→numer telefonu (`int`)). Posługiwanie się kontenerem `map` wygląda tak, jak gdybyśmy mieli tablicę, której elementy są indeksowane nie kolejnymi liczbami, a właśnie wartościami klucza. Kontener `map` przechowuje dla każdej wartości klucza tylko jedną wartość. Zarówno do dodawania jak i do dostępu do obiektów służy operator `[]`. Działanie jego jest jednak trochę inne niż dla wektora. Wywołanie operatora `[]` z jakimś kluczem powoduje zwrócenie referencji do obiektu stowarzyszonego z tym kluczem. Jeżeli w kontenerze nie ma takiego klucza, to kontener tworzy nową wartość stowarzyszoną z tym kluczem i zwraca referencję do tej nowej wartości. Wstawianie wartości i wyszukiwanie kluczy ma złożoność logarytmiczną względem wielkości kontenera. Aby użyć w kodzie źródłowym klasę `map` należy dołączyć odpowiedni plik nagłówkowy:

```
#include <map>
```

Obiekt klasy map odwzorowujący obiekty `string` na `int` tworzy się w następujący sposób:

```
map<string, int> nazwa_obiektu;
```

Bezpośrednio po utworzeniu, mapa nie zawiera żadnych obiektów. Poniżej kilka przydatnych metod z klasy `map`:

`[key]`	umożliwia dostęp do obiektu odpowiadającemu kluczowi `key`
`size()`	zwraca bieżącą ilość obiektów w mapie
`begin(), end()`	zwraca wskaźnik (iterator) do pierwszego lub ostatniego obiektu
`find(key)`	zwraca iterator do obiektu wskazywanego przez `key`
`count(key)`	zwraca ilość wystąpień klucza `key` w mapie
`erase(iterator position)`	usuwa obiekt na miejscu wskazywanym przez `position`
`clear()`	czyści zawartość mapy

Dla przykładu, dodanie nowej wartości 5 typu `int` dla klucza “ex” typu `string` będzie mieć postać:

```
nazwa_mapy[”ex”]=5;
```

Podobnie wykonuje się również modyfikację obiektów w mapie, np: `nazwa_mapy[”ex”]=7;`

Jeśli w mapie nie istnieje klucz to metoda `find(key)` zwraca iterator `end()`.

Poniżej przykład jak wyświetlić wszystkie obiekty w mapie:

```
map<string, int> mapOfWordCount;
// Iterate over the map using c++11 range based for loop
    for ( auto element : mapOfWordCount ) {
       string word = element.first; // Accessing KEY from element
       int count = element.second; // Accessing VALUE from element
       cout << word << " :: " << count << endl;
    }
```

1. Napisz program, który będzie prostą książką telefoniczną. Program działa w następujący sposób:
    a. W pętli powtarzaj:
        - Program prosi o wprowadzenie imienia osoby
            - Jeśli wprowadzone imię występuje już w książce to następuje wypisanie na ekranie numeru telefonu tej osoby.
            - Jeśli wprowadzone imię nie jest obecne w książce, to program prosi o podanie numeru telefonu i dodaje odpowiedni wpis do książki.
            - Jeśli jako imię wprowadzono wyraz “exit”, wyjdź z pętli i zakończ program.

2. Napisz program, który policzy i wypisze na ekranie częstość występowania liter we wprowadzonym z klawiatury stringu.

3. Napisz program, który wyświetli numery linii dla wszystkich wprowadzonych z klawiatury wyrazów. Przyjmij, że pierwszy wprowadzony wyraz znajduje się w linii numer 1. Program ma działać w następujący sposób:
    a. W pętli powtarzaj
        - wprowadź kolejny wyraz z klawiatury
            - Jeśli wprowadzono słowo “end”, zakończ wprowadzanie kolejnych wyrazów i wyświetl na ekranie numery linii w których występują wszystkie wprowadzone wyrazy

    Na przykład, dla danych wejściowych:
    ```
    car
    plane
    car
    ```
    program powinien wypisać na ekranie:
    ```
    car => 1 3
    plane => 2
    ```

4. DNS (ang. *Domain Name System*) to rozproszony system tłumaczący nazwy domen (np. twojadomena.pl) na adresy IP docelowych serwerów (np. 212.85.96.183). Gdy w przeglądarce wpisujemy nazwę domeny to najpierw wysyłane jest zapytanie do serwera DNS o adres IP dla wprowadzonej domeny. Gdy przeglądarka odbierze adres IP, to następuje połączenie z nim i otwierana jest wskazana strona internetowa. Napisz program, który będzie prostą bazą danych serwera DNS. Dla uproszczenia przyjmij, że jednej domenie internetowej odpowiada tylko jeden adres IPv4. Nazwy domen będą przechowywane w programie jako `string`, zaś adresy IP jako struktury postaci:
    ```
    struct IPv4{
        unsigned char o1;
        unsigned char o2;
        unsigned char o3;
        unsigned char o4;
    };
    ```
    Program powinien wczytać adresy IP odpowiadające domenom z pliku. Następnie powinien umożliwić zapytanie o adres IP domeny. Dodatkowo program powinien umożliwić wykonanie tak zwanego zapytania odwrotnego (ang. *reverse IP lookup*) czyli uzyskania domeny dla zadanego adresu IP.