Liczby słownie (rozwiązanie)

Autor zadania: **Karol Pokorski**
Opracowanie: **Krzysztof Kiljan, Michał Niedziółka**
Opis rozwiązania: **Lech Duraj**

Olimpiada Informatyczna Juniorów

To zadanie mocno różni się od pozostałych zadań OIJ – nie wymaga głębokiej znajomości matematyki, nie trzeba też w nim dbać o złożoność obliczeniową. Bardziej przypomina wyzwanie, przed którym mógłby stanąć „komercyjny" programista: jak zaprojektować program, by nie był przesadnie długi, pisanie go nie było zbyt żmudne, a przede wszystkim – żeby nie narobić trudnych do znalezienie błędów?

Takie zadania zawsze warto najpierw dobrze przemyśleć. Od razu widać, że w zadaniu będzie wiele różnych przypadków (np. czasem jest `tysiac`, czasem `tysiace`, a czasem wyrazu z tysiącami w ogóle nie ma; $20 + 7$ to `dwadziescia siedem`, $40 + 7$ to `czterdziesci siedem`, ale $10 + 7$ to `siedemnascie`...). Łatwymi pokusami jest choćby rozważanie takich przypadków za pomocą rozgałęzionych instrukcji warunkowych `if`, albo kilkukrotne kopiowanie tego samego kodu z drobnymi zmianami. To prawie zawsze niefortunne wybory – odkrywamy pod koniec, że zapomnieliśmy o jednym przypadku, albo znajdujemy błąd w wielokrotnie skopiowanej instrukcji. Dobrze jest zatem zaplanować swój kod tak, żeby był możliwie elastyczny, łatwy do modyfikacji, i żeby jak najmniej kodu się w nim powtarzało.

Pierwszą czynnością, którą powinniśmy wykonać, jest podzielenie napisu na poszczególne słowa. W języku Python służy do tego prosta instrukcja `split()`, w C++ najwygodniej użyć obiektu stringstream (ale można też wczytywać z wejścia poszczególne słowa po jednym za pomocą `cin/scanf`, albo zrobić pętlę iterującą się po znakach). W dalszej części będziemy zakładać, że mamy już listę/tablicę/wektor zawierający kolejne słowa wejściowego napisu.

Teraz zastanówmy się, jak wygląda liczba zapisana słownie – powinna mieć postać podobną do poniższej:

<ile> <miliony> <ile> <tysiące> <ile>

gdzie fragment *<miliony>* oznacza słowo *milion* w pewnej odmianie (`milion/miliony/milionow`), fragment *<tysiące>* to analogicznie `tysiac/tysiace/tysiecy`, zaś fragment *<ile>* składa się z 1-3 słów i oznacza liczbę między 1 a 999.

Na przykład:

$\underbrace{\texttt{sto trzydziesci piec}}_{<ile>}$ $\underbrace{\texttt{milionow}}_{<miliony>}$ $\underbrace{\texttt{szesnascie}}_{<ile>}$ $\underbrace{\texttt{tysiecy}}_{<tysiace>}$ $\underbrace{\texttt{dwiescie jeden}}_{<ile>}$.

(Osobno rozpatrujemy przypadek, który nie wpada w powyższy schemat, czyli napis `jeden miliard` – na szczęście jest tylko jeden taki).

Nasz algorytm tłumaczenia słownego napisu może więc działać następująco:

1. Podzielić napis na części odpowiadające fragmentom *<ile>*, *<tysiace>* i *<miliony>*.
2. Każdą z części typu *<ile>* przetłumaczyć na odpowiednią liczbę.
3. Zsumować wszystkie części i sprawdzić, czy zgadzają się kolejność i odmiana.

Nie unikniemy, niestety, zapamiętania w naszym programie wszystkich potrzebnych nam liczebników w języku polskim, na przykład w następującej formie:

liczebniki.cpp

```
string jednosci[]       =  { "jeden", "dwa", "trzy", "cztery", "piec", "szesc", "siedem"
    , "osiem", "dziewiec"}
string kilkanascie[]    =  { "dziesiec", "jedenascie", "dwanascie", "trzynascie", "
    czternascie", "pietnascie", "szesnascie", "siedemnascie", "osiemnascie", "
    dziewietnascie"};
string dziesiatki[]     = { "dwadziescia", "trzydziesci", "czterdziesci", "piecdziesiat"
    , "szescdziesiat", "siedemdziesiat", "osiemdziesiat", "dziewiecdziesiat"};
string setki[]          = {"sto", "dwiescie", "trzysta", "czterysta", "piecset", "
    szescset", "siedemset", "osiemset", "dziewiecset"};
string tysiace[]        = {"tysiac", "tysiace", "tysiecy"};
string miliony[]        = {"milion", "miliony", "milionow"};
```

(kod w Pythonie pomijamy, jako że różni się wyłącznie rozmieszczeniem nawiasów i średników)

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

Aby zrealizować punkt 1., iterujemy się po słowach z wejścia: jeśli kolejne słowo jest jednym z tablicy `tysiace[]` lub `miliony[]`, tworzy osobny fragment, w innym wypadku zakładamy nowy fragment typu *<ile>*, lub dołączamy słowo do poprzedniego fragmentu.

W punkcie 2., tłumacząc fragment typu *<ile>*, sprawdzamy uważnie, czy nasz zbiór słów spełnia odpowiednie warunki:

- wszystkie słowa odpowiadają zapamiętanym przez nas liczebnikom;
- zawiera najpierw co najwyżej jedną setkę (element tablicy `setki[]`), a potem albo jeden element `kilkanascie[]`, albo co najwyżej jeden element `dziesiatki[]` i co najwyżej jeden `jednosci[]`.

Jeżeli fragment nie dopasowuje się do tego schematu, liczba jest niepoprawna. Jeśli zaś uda się dopasować, tłumaczymy liczebniki na odpowiednie wartości (element `setki[i]` tłumaczy się na $100 \cdot (i+1)$, element `kilkanascie[i]` na $10+i$, itd.) i sumujemy.

Alternatywną metodą tłumaczenia fragmentu *<ile>* może być sprawdzenie kolejno wszystkich liczb od 1 do 999, stworzenie dla każdej słownego zapisu i sprawdzenie, która liczba tłumaczy się dokładnie na nasz fragment. Ograniczenia czasowe zadania pozwalają na swobodne sprawdzenie wszystkich możliwości.

Na koniec dopasowujemy fragmenty do liczebników tysiąca/miliona i sprawdzamy, czy są w odpowiedniej kolejności (miliony, tysiące, jedności), czy każdego rodzaju jest co najwyżej jeden (fragment *<tysiac>* nie może występować więcej niż raz – choć oczywiście może go nie być w ogóle) i czy zgadza się odmiana (jeśli fragment `<ile>` kończył się na `jeden`, powinno być `tysiac/milion`, jeśli na `dwa`, `trzy` lub `cztery` – `tysiace/miliony`, w innym wypadku `tysiecy/milionow`).

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".