4. Czy to jest palindrom?

Gry i zabawy słowne są znane i lubiane od wieków. Być może zdarzyło ci się grać w grę polegającą na szukaniu wyrazu zaczynającego się na ostatnią literę wyrazu podanego przez poprzednią osobę. Znasz też na pewno wiele tzw. łamańców językowych, czyli wyrażeń tworzonych celowo tak, żeby trudno je było wymówić, jak „stół z powyłamywanymi nogami". Inną zabawą, która od starożytności zajmuje ludzi posługujących się różnymi językami, jest szukanie słów, a nawet całych zdań, które brzmią tak samo czytane wspak. Właśnie takimi wyrażeniami zajmiemy się w tym temacie.

Cele lekcji

- Dowiesz się, czym jest palindrom.
- Zbadasz, czy wyraz oraz zdanie są palindromami.
- Wykonasz operacje na zmiennych napisowych.
- Zdefiniujesz własne funkcje w języku C++.
- Poznasz różnicę między parametrem formalnym i aktualnym.
- Wyszukasz w zdaniu wyrazy będące palindromami.

Palindrom to słowo, wyrażenie bądź zdanie, które czytane od lewej do prawej i od prawej do lewej brzmi tak samo. Nazwa wywodzi się od greckiego słowa *palìndromos*, które znaczy „biegnący na powrót". Palindromami są np. słowo „sms" czy wyrażenie „oko w oko".

Palindrom

4.1. Sprawdzamy, czy wyraz jest palindromem

Pierwszym zadaniem, które sobie postawimy, będzie sprawdzenie, czy pojedynczy wyraz jest palindromem. Oto specyfikacja problemu:

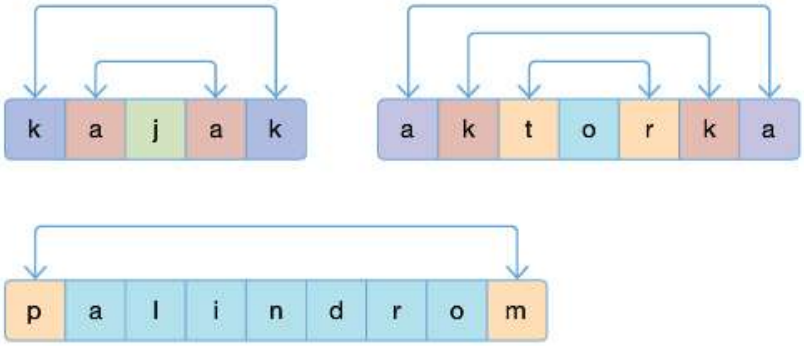
Specyfikacja

Dane: `wyraz` – słowo złożone z małych liter alfabetu łacińskiego.

Wynik: słowo „TAK", gdy wyraz jest palindromem, słowo „NIE" – w przeciwnym przypadku.

Podczas sprawdzania będziemy porównywać kolejne pary liter: pierwszą z ostatnią, drugą z przedostatnią itd. Znalezienie pierwszej pary różnych liter oznacza, że wyraz nie jest palindromem, więc można wtedy zakończyć analizę. Jeśli nie znajdziemy pary różnych liter, to wyraz jest palindromem. Zauważ, że jeśli palindrom składa się z nieparzystej liczby liter, nie musimy porównywać środkowej litery samej ze sobą. Ogólnie, gdy wyraz jest palindromem, wykonamy n **`div`** 2 porównań par liter, gdzie n oznacza liczbę liter w wyrazie.

Rysunek 4.1 przedstawia działanie algorytmu dla trzech przykładowych wyrazów. Pierwszy wyraz jest palindromem – tu trzeba wykonać dwa porównania. Kolejne dwa wyrazy nie są palindromami. Aby się o tym przekonać, w przypadku drugiego wyrazu musimy wykonać trzy porównania, a w przypadku trzeciego – jedno porównanie.



Rys. 4.1. Sprawdzenie, czy podane wyrazy są palindromami

Dobra rada

Podany algorytm zadziała dla dowolnego ciągu znaków bez spacji. Za jego pomocą możesz więc np. sprawdzić, czy liczba reprezentowana jako napis jest palindromem.

W pseudokodzie zapis algorytmu sprawdzającego, czy wyraz jest palindromem, może wyglądać następująco:

```
palindrom ← prawda
i ← 0
j ← długość wyrazu - 1
dopóki palindrom oraz (i < j) wykonuj
    jeśli wyraz[i] = wyraz[j] to
        i ← i + 1
        j ← j - 1
    w przeciwnym przypadku palindrom ← fałsz
jeśli palindrom to wypisz "TAK"
w przeciwnym przypadku wypisz "NIE"
```

Na początku zakładamy, że badany wyraz jest palindromem, dlatego zmiennej `palindrom` typu logicznego przypisujemy wartość **prawda**. Dodatkowo przyjmujemy, że słowo puste bądź słowo złożone tylko z jednej litery jest palindromem. Zmienna `i` wskazuje kolejne litery z początku wyrazu (wartość początkowa tej zmiennej jest równa 0, ponieważ znaki napisu indeksowane są od zera), a zmienna `j` – kolejne litery od końca wyrazu (wartość początkowa jest równa indeksowi ostatniej litery wyrazu, czyli długości wyrazu pomniejszonej o 1). Jeśli w aktualnie porównywanej parze są takie same litery, wartość zmiennej `i` jest powiększana o 1, a zmiennej `j` – pomniejszana o 1. W przypadku, gdy litery w badanej parze są różne, zmienna `palindrom` przyjmuje wartość **fałsz**, co powoduje zakończenie wykonywania pętli. Jeśli badany wyraz jest palindromem, instrukcje wykonują się, dopóki `i < j`.

Zapiszemy teraz kod źródłowy programu realizującego podany algorytm. Program zadziała błędnie, gdy wyraz będzie zawierał zarówno małe, jak i wielkie litery, np. litery „A" i „a" odczyta jako różne znaki.

Kod źródłowy programu sprawdzającego, czy słowo jest palindromem

```
1.  #include <iostream>
2.  #include <string>
3.
4.  using namespace std;
5.
6.  int main()
7.  {
8.      string wyraz;
9.      int i=0, j;
10.     bool palindrom=true;
11.     cout<<"Podaj wyraz: "; cin>>wyraz;
12.     j=wyraz.size()-1;
13.     while (palindrom && i<j)
14.         if (wyraz[i]==wyraz[j])
15.         {
16.             i++;
17.             j--;
18.         }
19.         else palindrom=false;
20.     if (palindrom) cout<<"TAK";
21.     else cout<<"NIE";
22.     return 0;
23. }
```

Aby program wypisywał słowo „TAK" niezależnie od tego, czy do zapisu palindromu użyto małych czy wielkich liter, możemy np. zastosować **funkcję** `toupper`. Jej wartością jest kod ASCII wielkiej litery, jeśli **parametr** był małą literą. W przeciwnym przypadku wartością funkcji jest parametr bez zmiany. Na przykład dla parametrów `'A'` i `'a'` wynikiem funkcji będzie ta sama wartość – liczba 65 (kod ASCII litery „A"). W kodzie źródłowym programu wystarczy zmienić warunek w instrukcji warunkowej (linia 14) na:

Funkcja `toupper`

Parametr funkcji, s. 64

```
toupper(wyraz[i])==toupper(wyraz[j])
```

Zamiast funkcji `toupper` moglibyśmy skorzystać z **funkcji** `tolower`. Jej wartością jest kod ASCII małej litery, jeśli parametr był wielką literą. W przeciwnym przypadku wartością funkcji jest parametr bez zmiany, podobnie jak w przypadku funkcji `toupper`.

Funkcja `tolower`

Funkcje `toupper` i `tolower` są zdefiniowane w bibliotece `cctype`.

Ćwiczenie 1

a. Napisz program sprawdzający, czy wyraz złożony z małych liter alfabetu łacińskiego jest palindromem.

b. Zmodyfikuj specyfikację oraz program sprawdzający, czy wyraz jest palindromem, w taki sposób, aby wypisywał słowo „TAK" niezależnie od tego, czy do zapisu wyrazu użyto małych czy wielkich liter.

4.2. Znajdujemy słowa palindromy w zdaniu

Rozwiążemy teraz kolejny problem – sprawdzimy, które wyrazy w zdaniu są palindromami. Program, który napiszemy, wypisze palindromy – niezależnie od tego, czy do zapisu użyto małych czy wielkich liter. Zakładamy, że jedynymi separatorami słów w zdaniu są spacje.

Specyfikacja

Dane: zdanie – napis złożony z liter alfabetu łacińskiego oraz spacji.

Wynik: słowa palindromy z napisu zdanie.

Zauważ, że zwykle w zdaniu spacja występuje po każdym słowie z wyjątkiem ostatniego. Jeśli dodamy spację na końcu napisu, wówczas po każdym wyrazie będzie spacja. Dodany znak będzie wtedy pełnić funkcję **wartownika**, czyli elementu ułatwiającego odnalezienie końca danych – w naszym przypadku końca zdania.

Wartownik

Algorytm znajdujący palindromy w zdaniu będzie działał następująco: wyszukujemy pierwsze wystąpienie znaku spacji w zdaniu, a następnie badamy, czy wyraz złożony z liter znajdujących się przed tą spacją jest palindromem. Po sprawdzeniu usuwamy ten wyraz ze zdania. Wymienione czynności powtarzamy, dopóki napis nie jest pusty.

Do zapamiętywania słów użyjemy zmiennej `wyraz`. Jeśli w danej iteracji pętli wartość zmiennej będzie palindromem, program wypisze ją na ekranie. Oto zapis algorytmu w pseudokodzie:

Warto wiedzieć

Jeśli pierwszy znak napisu jest spacją, to wykona się tylko instrukcja po instrukcji warunkowej. Spowoduje to usunięcie spacji z początku zdania.

```
zdanie ← zdanie + ' '
dopóki długość zdania > 0 wykonuj
    i ← miejsce wystąpienia pierwszej spacji
    jeśli i > 0 to // jeśli spacja nie jest pierwszym znakiem
        wyraz ← pierwsze i znaków zdania
        jeśli Palindrom(wyraz) to wypisz wyraz
    zdanie ← zdanie bez i + 1 początkowych znaków
    // usuń ze zdania wyraz razem ze spacją
```

W podanym algorytmie założyliśmy, że mamy do dyspozycji operację, która sprawdza, czy wyraz jest palindromem – nazwaliśmy ją `Palindrom`. Dzięki rozłożeniu problemu algorytmicznego na podproblemy algorytm jest bardziej czytelny.

Funkcje w języku C++

W praktyce rozkładanie problemów na mniejsze oznacza wyodrębnianie fragmentów programów, czyli podprogramów, które w C++ nazywamy **funkcjami**. Często korzystamy z różnych funkcji (dostępnych w bibliotekach), choć nie znamy ich kodu ani algorytmów, które wykonują. Musimy wiedzieć jedynie, jakie dane, nazywane **parametrami**, należy dostarczyć oraz co jest wynikiem danej funkcji.

Funkcja

Parametr funkcji

Budowa funkcji w języku C++ jest analogiczna do budowy znanej ci funkcji `main`. **Definicja funkcji** składa się z nagłówka funkcji i bloku instrukcji. Nagłówek zawiera typ funkcji, jej nazwę oraz listę parametrów, tzn. wskazanie typu i nazwy każdego z nich.

Definicja funkcji

```
typ nazwa_funkcji(typ parametr1, typ parametr2, ...)
{
    // blok instrukcji funkcji
}
```

Parametry określone w definicji funkcji nazywamy **parametrami formalnymi**. Natomiast parametry, dla których funkcja jest wywołana, to **parametry aktualne**. Podczas wywołania funkcji w miejsce parametrów formalnych podstawiane są parametry aktualne. Typy parametrów aktualnych muszą być zgodne z typami parametrów formalnych.

Parametr formalny

Parametr aktualny

Własne funkcje w języku C++ definiujemy przed funkcją `main`. Jeśli funkcja ma zwrócić wartość, w bloku instrukcji musi zostać użyta instrukcja `return` – po niej podajemy zwracaną wartość zgodną z typem funkcji określonym w nagłówku. Instrukcja `return` jednocześnie kończy działanie funkcji. Istnieją także funkcje niezwracające wartości – są to **funkcje typu `void`**. Poznasz je w temacie 7.

Funkcja typu `void`, s. 113

Dzięki konsekwentnemu stosowaniu funkcji tworzony kod jest bardziej przejrzysty i łatwiejszy do modyfikowania. Unikamy także powtórzeń, ponieważ tę samą funkcję można wywołać wielokrotnie dla różnych wartości parametrów.

Zapamiętaj

Definicja funkcji składa się z dwóch części: nagłówka oraz bloku instrukcji, które funkcja ma wykonać. W nagłówku podaje się typ funkcji, jej nazwę oraz listę parametrów formalnych, zawierającą nazwę i typ każdego z parametrów. Jeśli funkcja ma zwrócić wartość, należy użyć instrukcji `return`, a po niej podać wartość, która będzie wynikiem funkcji.

Definicja funkcji `Palindrom`

Teraz zdefiniujemy funkcję `Palindrom`, którą wykorzystamy do rozwiązania problemu głównego – znajdowania palindromów w zdaniu. Parametrem funkcji jest napis bez spacji (wyraz), a wynikiem – wartość logiczna informująca, czy wyraz jest palindromem czy nie. Funkcja `Palindrom` musi zwrócić wartość logiczną, ponieważ w algorytmie rozwiązującym problem główny używamy tej funkcji w warunku instrukcji warunkowej (szósta linia pseudokodu na poprzedniej stronie). W kodzie źródłowym programu definicja funkcji znajduje się w liniach 7–18.

I część kodu źródłowego programu wypisującego słowa palindromy ze zdania – definicja funkcji `Palindrom`

```
1.  #include <iostream>
2.  #include <string>
3.  #include <cctype>
4.
5.  using namespace std;
6.
7.  bool Palindrom(string wyraz)
8.  {
9.      int i=0, j=wyraz.size()-1;
10.     bool p=true;
11.     while (p && i<j)
12.         if (toupper(wyraz[i])==toupper(wyraz[j]))
13.         {
14.             i++; j--;
15.         }
16.         else p=false;
17.     return p;
18. }
```

W nagłówku funkcji (linia 7) są określone typ wyniku funkcji: **bool** oraz nazwa funkcji: `Palindrom`, a w nawiasach – parametr o nazwie `wyraz` typu **string**. Zwróć uwagę na instrukcję w linii 17. Ponieważ wynikiem funkcji jest wartość typu logicznego, instrukcja **return** zwraca wartość zmiennej `p`, w której przechowywana jest wartość logiczna informująca, czy wyraz jest palindromem.

Definicja funkcji `main` wykorzystującej funkcję `Palindrom`

Definicja funkcji `main` realizującej algorytm zapisany w pseudokodzie na s. 64, wykorzystujący funkcję `Palindrom`, może być następująca:

II część kodu źródłowego programu wypisującego słowa palindromy ze zdania – funkcja `main`

```
19. int main()
20. {
21.     int i;
22.     string wyraz, zdanie;
23.     cout<<"Podaj zdanie: ";
24.     getline(cin,zdanie);
25.     zdanie=zdanie+' ';
26.     while (zdanie.size()>0)
27.     {
28.         i=zdanie.find(' ');
29.         if (i>0)
30.         {
31.             wyraz=zdanie.substr(0,i);
32.             if (Palindrom(wyraz)) cout<<wyraz<<endl;
33.         }
34.         zdanie.erase(0,i+1);
35.     }
36.     return 0;
37. }
```

Aby podane przez użytkownika zdanie wczytać do programu, użyliśmy **funkcji** `getline` (linia 24). Pierwszym parametrem tej funkcji jest miejsce, z którego pobieramy dane – w naszym przypadku jest to klawiatura, dlatego parametrem jest `cin`. Drugim parametrem funkcji jest zmienna, w której zapiszemy dane. Gdybyśmy do wczytania danych użyli instrukcji `cin>>zdanie`, program zapisałby tylko fragment napisu do wystąpienia pierwszej spacji. Funkcja `getline` wczytuje cały wiersz danych.

Funkcja `getline`

Po wczytaniu zdania na końcu dodawana jest spacja – **wartownik**. Pętla dzieląca zdanie na wyrazy (linie 26–35) korzysta z metod operujących na napisach, dostępnych w klasie `string`: `size`, `find`, `substr`, `erase`. O **metodzie** `size` mówiliśmy w temacie 2. Informacje o **metodach** `find`, `substr`, `erase` znajdziesz w tabeli 4.1. Należy pamiętać, że przy stosowaniu metod obowiązuje notacja z kropką, tzn. `nazwa_zmiennej.nazwa_metody`.

Wartownik, s. 64

Metoda `size`, s. 39

Metody `find`, `substr`, `erase`

Metoda	Parametry	Wynik	Przykład zastosowania metody dla zmiennej `s` typu `string` `s="Ala ma kota"`

`find(szukane)`	`szukane` – fraza lub znak	Pozycja pierwszego wystąpienia szukanej frazy w napisie. Jeśli dany fragment nie występuje, wynikiem metody jest liczba spoza zakresu `0...size()-1`.	Instrukcja:
			`cout<<s.find("ma");`
			Wynik:
			`4`

`substr(pozycja,ile)`	`pozycja` – indeks znaku w napisie, `ile` – liczba znaków	Fragment ciągu znaków zapisanych od indeksu `pozycja`. Liczbę znaków określa parametr `ile`.	Instrukcja:
			`cout<<s.substr(4,2);`
			Wynik:
			`ma`

`erase(pozycja,ile)`	`pozycja` – indeks znaku w napisie, `ile` – liczba znaków	Napis bez ciągu znaków o długości `ile`, liczonego od indeksu `pozycja`.	Instrukcje:
			`s.erase(4,3);`
			`cout<<s;`
			Wynik:
			`Ala kota`

Tabela 4.1. Parametry i wyniki metod `find`, `substr`, `erase` oraz przykłady ich zastosowania

W linii 28 poszukiwane jest pierwsze wystąpienie znaku spacji. Jeśli nie jest to pierwszy znak zdania (a więc na lewo od spacji znajdują się litery tworzące wyraz), to ten wyraz jest zapamiętywany w zmiennej `wyraz` (linia 31). Jeśli wyraz jest palindromem, to zostanie wypisany na ekranie (linia 32). Ostatnia instrukcja w pętli (linia 34) odpowiada za usunięcie ze zdania zbadanego wyrazu wraz ze spacją (lub samej spacji, gdy jest ona pierwszym znakiem napisu).

Ćwiczenie 2

Napisz program znajdujący wyrazy palindromy w zdaniu wpisanym z klawiatury (niezależnie od tego, czy w zapisie użyto małych czy wielkich liter). W programie zdefiniuj funkcję sprawdzającą, czy wyraz jest palindromem.

Zapamiętaj

Aby wczytać do programu wiersz danych, w których znajdują się spacje, należy użyć funkcji `getline`. Funkcja ta wymaga podania dwóch parametrów. Pierwszy określa miejsce, z którego pobieramy dane, natomiast drugim jest zmienna, w której zapiszemy dane. Na przykład do zapisania w zmiennej `s` zdania podanego z klawiatury zastosujemy instrukcję `getline(cin,s)`. Instrukcja `cin>>s` wczyta dane do znaku spacji lub znaku końca wiersza.

Modyfikacja funkcji `Palindrom`

W definicji funkcji `Palindrom` nie musimy używać zmiennej pomocniczej `p` typu logicznego. Zamiast niej można zastosować wielokrotnie instrukcję **`return`**.

Zmodyfikowana funkcja `Palindrom`

```
7.  bool Palindrom(string wyraz)
8.  {
9.      int i=0, j=wyraz.size()-1;
10.     while (i<j)
11.         if (toupper(wyraz[i])==toupper(wyraz[j]))
12.         {
13.             i++; j--;
14.         }
15.         else return false;
16.     return true;
17. }
```

Jeśli wyraz jest palindromem, to wynikiem funkcji jest wartość **`true`** (linia 16). Natomiast gdy znaki w porównywanej parze okażą się różne (warunek logiczny w linii 11 przyjmie wartość **`false`**), funkcja zakończy działanie, a jej wynikiem będzie **`false`** (linia 15).

Ćwiczenie 3

Program znajdujący wyrazy palindromy w zdaniu zmodyfikuj tak, aby w funkcji sprawdzającej, czy wyraz jest palindromem, nie była użyta zmienna pomocnicza typu logicznego.

4.3. Sprawdzamy, czy zdanie jest palindromem

Teraz napiszemy program, który sprawdzi, czy całe zdanie jest palindromem. Przykładami tego typu palindromów w języku polskim są zdania „Kobyła ma mały bok" i „A to kanapa pana kota".

Warto wiedzieć

Zdanie „Kobyła ma mały bok" uchodzi obecnie za najbardziej znany polski palindrom. Pochodzi z XIX w. i nie wiadomo, kto jest jego autorem.

Specyfikacja

Dane: `zdanie` – napis złożony z liter alfabetu łacińskiego, spacji oraz znaków przestankowych.

Wynik: słowo „TAK", jeśli `zdanie` jest palindromem, słowo „NIE" – w przeciwnym przypadku.

Żeby zbadać, czy zdanie jest palindromem, usuniemy z napisu wszystkie znaki inne niż litery. Dzięki temu, podobnie jak w programie badającym, czy wyraz jest palindromem, sprawdzimy, czy napis złożony z samych liter jest palindromem. Na przykład dla napisu „Kobylamamalybok" program wypisze słowo „TAK".

W kodzie podanym poniżej w liniach 7–14 zdefiniowano funkcję `TylkoLitery`, której parametrem jest zdanie, a wynikiem – napis złożony z samych liter (funkcja usuwa znaki inne niż litery). W dalszych liniach znajduje się kod źródłowy funkcji `main`, która wypisuje informację, czy zdanie jest palindromem.

Kod źródłowy programu sprawdzającego, czy zdanie jest palindromem

```
1.  #include <iostream>
2.  #include <string>
3.  #include <cctype>
4.
5.  using namespace std;
6.
7.  string TylkoLitery(string s)
8.  {
9.      int i=0;
10.     while (i<s.size())
11.         if (toupper(s[i])>='A' && toupper(s[i])<='Z') i++;
12.         else s.erase(i,1);
13.     return s;
14. }
15.
16. int main()
17. {
18.     string zdanie, wyraz;
19.     int i=0, j;
20.     bool palindrom=true;
21.     cout<<"Podaj zdanie: "; getline(cin,zdanie);
22.     wyraz=TylkoLitery(zdanie);
23.     j=wyraz.size()-1;
```

Dobra rada

Do sprawdzenia, czy znak jest literą, możesz też wykorzystać funkcję `isalpha` z biblioteki `cctype`.

Dobra rada

Pamiętaj, że kompilator języka C++ rozróżnia wielkie i małe litery. Jeśli deklarujesz zmienną lub funkcję zawierającą w nazwie wielkie litery, to w dalszej części kodu musisz konsekwentnie używać ustalonej nazwy.

```
24.     while (palindrom && i<j)
25.         if (toupper(wyraz[i])==toupper(wyraz[j]))
26.         {
27.             i++; j--;
28.         }
29.         else palindrom=false;
30.     if (palindrom) cout<<"TAK";
31.     else cout<<"NIE";
32.     return 0;
33. }
```

W pętli `while` (linie 10–12) w funkcji `TylkoLitery` badane są kolejne znaki napisu. Jeśli aktualny znak jest małą lub wielką literą alfabetu łacińskiego, to powiększany jest indeks `i` (linia 11). W przeciwnym przypadku znak usuwany jest z napisu (linia 12). Wykorzystano do tego **metodę** `erase`, która usuwa z danego napisu określoną liczbę znaków (drugi parametr), zaczynając od konkretnej pozycji (pierwszy parametr). Zdanie do programu wczytuje **funkcja** `getline` (linia 21), podobnie jak w programie znajdującym słowa palindromy w zdaniu.

Metoda `erase`, s. 67

Funkcja `getline`, s. 67

Ćwiczenie 4

Napisz program sprawdzający, czy zdanie jest palindromem. Sprawdź działanie programu dla różnych zdań. Przykładów palindromów możesz poszukać w internecie.

A to ciekawe

Palindromy w kulturze i sztuce

Wymyślanie palindromów traktuje się najczęściej jako zabawę, choć mogą one służyć do przekazania jakiejś ważnej myśli. Zdjęcie obok przedstawia część fontanny z jednym z najstarszych znanych palindromów. Grecki napis oznacza „zmyj grzechy, nie tylko twarz". Za najstarszy polski palindrom uważa się pochodzące z XVII w. zdanie autorstwa księdza Wojciecha Waśniowskiego: „Co w onei? ow pokoy y okop, woien owoc". Palindromami zajmowali się także polscy poeci, m.in. Julian Tuwim, który w książce *Pegaz dęba* zebrał wiele różnych gier słownych, w tym palindromów. Tadeusz Morawski, profesor Politechniki Warszawskiej, jest czołowym polskim twórcą palindromów. Wydał kilkanaście książek je zawierających, założył też Muzeum Palindromów w Nowej Wsi koło Serocka.

Podsumowanie

- Palindrom to słowo, wyrażenie lub zdanie, które brzmi tak samo czytane od lewej do prawej i od prawej do lewej.
- Definicja funkcji w języku C++ składa się z dwóch głównych części: nagłówka funkcji oraz bloku instrukcji. W nagłówku podajemy typ funkcji, jej nazwę oraz listę parametrów formalnych (lista zawiera typ i nazwę każdego z parametrów).
- Parametry formalne to parametry podane w definicji funkcji.
- Parametry aktualne to parametry, dla których funkcja jest wywołana.
- Typy parametrów aktualnych muszą być zgodne z typami parametrów formalnych.
- Zapisanie podproblemu w postaci funkcji poprawia czytelność programu i umożliwia obliczanie wartości funkcji dla różnych wartości parametrów.
- Wynikiem funkcji `toupper` dla parametru będącego literą jest kod ASCII wielkiej litery, natomiast wynikiem funkcji `tolower` dla parametru będącego literą jest kod ASCII małej litery.
- Specjalny element ułatwiający odnalezienie końca danych nazywamy wartownikiem.
- Aby wczytać wiersz danych ze spacjami, należy użyć funkcji `getline`.
- Dla zmiennych typu `string` dostępnych jest wiele użytecznych metod umożliwiających pracę z napisami, np. `size` – badanie długości napisu, `substr` – znajdowanie fragmentu napisu, `erase` – usuwanie fragmentu napisu, `find` – znajdowanie pierwszego wystąpienia fragmentu napisu.

Zadania

* **1** Napisz program, który wczyta napis i sprawdzi, czy składa się on wyłącznie z małych liter alfabetu łacińskiego. Program powinien wypisać słowo „TAK" lub „NIE".

* **2** Napisz program wczytujący zdanie złożone z liter alfabetu łacińskiego oraz spacji i wypisujący liczbę wyrazów w zdaniu.

* **3** Napisz program, który po wczytaniu zdania złożonego z liter alfabetu łacińskiego oraz spacji wypisze zdanie, w którym pierwsze litery każdego z wyrazów będą wielkie, a pozostałe litery w wyrazach – małe.

** **4** Napisz program, który po wczytaniu zdania złożonego z liter alfabetu łacińskiego oraz spacji sprawdzi, czy wszystkie wyrazy w tym zdaniu składają się z tej samej liczby liter oraz czy wszystkie są palindromami. Jeśli oba warunki będą spełnione, program wypisze słowo „TAK", w przeciwnym przypadku – słowo „NIE".

** **5** Napisz program, który wypisze wszystkie trzycyfrowe liczby palindromiczne.
Uwaga: Liczba palindromiczna to liczba naturalna, która czytana od lewej do prawej i od prawej do lewej ma taką samą wartość.