Palindromy w języku C++

Źródło: Pixabay, domena publiczna.

Palindromeo znaczy po grecku „biec z powrotem”. Palindromem nazywamy zatem wyrażenie, które brzmi tak samo czytane od strony lewej do prawej, jak i od prawej do lewej („a to kanapa pana kota”, „a tu mam mamuta” itp.). Podstawowe informacje na ten temat znajdziesz w e-materiale Palindromy.

Przygotowanie algorytmu służącego do sprawdzania, czy jakieś wyrażenie jest palindromem, często pojawia się wśród zadań na egzaminach z informatyki. W tym e-materiale zajmiemy się implementacją omawianego algorytmu w języku C++.

Implementację programu sprawdzającego, czy dane słowo jest palindromem w pozostałych językach programowania znajdziesz w e-materiałach:

- Palindromy w języku Java,
- Palindromy w języku Python.

Więcej zadań? Przejdź do e-materiału Palindromy – zadania maturalne.

> **Twoje cele**
>
> - Przeanalizujesz napisany w C++ program sprawdzający, czy dany ciąg znaków jest palindromem.
> - Porównasz różne sposoby sprawdzania, czy ciąg znaków jest palindromem.
> - Zaimplementujesz algorytm tworzenia palindromu.

Przeczytaj

Jak sprawdzić, czy dany ciąg liter lub cyfr jest palindromem, czyli czy wyrażenie odczytywane w obie strony brzmi tak samo?

Będziemy zajmować się ciągami znaków, a zatem skorzystamy z biblioteki:

```
1 #include <string>
```

Zapiszemy również wykorzystanie przestrzeni nazw:

```
1 #include <string>
2 using namespace std;
```

Dzięki wykorzystaniu biblioteki `<string>` możemy wykonywać różne działania na ciągach znaków, np. porównywanie, dodawanie kilku łańcuchów do siebie, wyszukanie konkretnego znaku w łańcuchu.

Aby zadeklarować ciąg znaków, używamy `string nazwa_zmiennej.`

> **Dla zainteresowanych**
>
> Czasami kompilatory dołączają niektóre biblioteki automatycznie, bez potrzeby deklaracji ich w kodzie. Np. po dołączeniu `<iostream>` może się okazać, że już jest w niej zawarty nagłówek `<string>`. Jednak, dla pewności, w implementacji należy dodawać niezbędne biblioteki. Program bowiem może nie działać przy użyciu innego kompilatora lub w innej wersji kompilatora przez ciebie używanego.

Odwracanie wyrażenia w łańcuchu znaków

Załóżmy, że chcemy sprawdzić, czy słowo „kajak” jest palindromem. Jednym ze sposobów badania ciągu znaków będzie odwrócenie go i porównanie z oryginałem.

Zaczniemy od zdefiniowania łańcucha znaków `string`, któremu nadamy nazwę `tekst`. Przypiszemy mu wartość `kajak`. Utworzymy również pusty łańcuch `odwrot`. Znajdzie się w nim sprawdzane wyrażenie, zapisane wspak.

```
string tekst = "kajak";
string odwrot;
```

Następnie użyjemy pętli `for` w celu wprowadzenia do zmiennej `odwrot` znaków łańcucha `tekst` zapisanych w odwrotnej kolejności.

Licznikowi w pętli `for` (zmiennej `i`) przypiszemy wartość ostatniego indeksu łańcucha znaków `tekst`. Wynosi ona `tekst.size() - 1`, ponieważ indeksowanie łańcucha zaczynamy od liczby `0` (słowo `kajak` składa się z pięciu znaków ponumerowanych od `0` do `4`). W każdym przejściu pętli będziemy dekrementować zmienną `i`, aż do momentu, gdy osiągnie ona wartość `0` (czyli taką samą, jak indeks pierwszego znaku łańcucha tekstowego).

```
for (int i = tekst.size() - 1; i >= 0; i--)
{
    odwrot += tekst[i];
}
```

Poszczególne elementy łańcucha znaków `odwrot` wprowadzamy za pomocą operatora przypisania `+=`.

Gdy gotowe są już dwa łańcuchy znaków – jeden ze sprawdzanym wyrażeniem, a drugi z jego lustrzanym odbiciem – możemy porównać je za pomocą instrukcji warunkowej `if`. Jeśli okażą się one identyczne, mamy do czynienia z palindromem.

```
if (tekst == odwrot)
    cout << "Palindrom";
else
    cout << "To nie jest palindrom";
```

Po zbadaniu słowa „kajak” otrzymalibyśmy odpowiedź `Palindrom`.

Porównywanie znaków o skrajnych indeksach

Inna metoda badania, czy wyrażenie jest palindromem, polega na porównywaniu jego skrajnych znaków. Ponieważ palindrom brzmi tak samo czytany w obie strony, to pierwszy znak jest taki sam jak ostatni, drugi taki sam jak przedostatni itd.

Znowu zaczniemy od zdefiniowania wyrażenia, które chcemy zbadać. Tworzymy łańcuch znaków `string` o nazwie `tekst` i wpisujemy do niego słowo `rower`. Definiujemy również zmienną typu logicznego (`bool`):

```
string tekst = "rower";
bool palindrom = true;
```

Zmienną logiczną nazwaliśmy `palindrom` – przechowywana w niej będzie informacja o tym, czy wyrażenie jest palindromem. Wstępnie nadaliśmy jej wartość `true`. W przypadku, gdy znaki o skrajnych indeksach nie będą sobie równe, zmienimy ją na `false`. Innymi słowy, początkowo zakładamy, że sprawdzany tekst jest palindromem, a następnie weryfikujemy, czy mieliśmy rację.

Następnie użyjemy pętli `for`. Posłuży ona do badania znaków na obu końcach wyrażenia. Zadeklarujemy dwie zmienne iteracyjne (licznikowe). Początkową wartością jednej z nich będzie indeks pierwszego znaku (`0`), natomiast wartość drugiej wstępnie ustalimy jako `tekst.size() - 1` (odpowiada to indeksowi ostatniego znaku wyrażenia).

```
for (int i = 0, j = tekst.size() - 1; i < tekst.size() / 2; i++,
{
    if (tekst[i] != tekst[j]) palindrom = false;
}
```

Pętla będzie wykonywana dopóty, dopóki prawdziwy będzie warunek `i < tekst.size() / 2`. Znaki sprawdzamy parami, a zatem liczba

powtórzeń pętli odpowiada połowie długości kontrolowanego łańcucha. Zmienna `palindrom` przyjmie wartość zależną od tego, czy mamy do czynienie z palindromem, czy też nie.

Zmienne logiczne mogą służyć za wyrażenie sterujące działaniem funkcji warunkowej `if`. Uzupełnimy więc kod:

```
1 if (palindrom) cout << "Palindrom";
2 else cout << "To nie jest palindrom";
```

Po zbadaniu słowa „rower” otrzymamy odpowiedź: `To nie jest palindrom`.

Słownik

> **dekrementacja**
>
> zmniejszenie wartości zmiennej o jeden

> **palindrom**
>
> słowo lub wyrażenie, które brzmi tak samo odczytywane zarówno od strony lewej do prawej, jak i w odwrotnym kierunku

Symulacja interaktywna

Polecenie 1

Zapoznaj się z prezentacją, następnie wykonaj ćwiczenie.

Ćwiczenie 1

Napisz program, którego celem będzie sprawdzenie, czy dany łańcuch znaków *test* jest palindromem. Swój program przetestuj dla następującego ciągu znaków:

```
Może jutro ta dama sama da tortu jeżom
```

Do testu wykorzystaj symulację interaktywną.

Specyfikacja problemu:

Dane:

- *test* – łańcuch znaków o długości $n \geq 0$

Wynik:

Jeśli łańcuch znaków jest palindromem, program wyświetla komunikat *Palindrom* lub *To nie jest palindrom* w przeciwnym wypadku.

Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/a/D2KF7ip0f

Polecenie 2

Dodaj do swojego programu komentarze tak, żeby był zrozumiały dla osoby, która nie potrafi programować.

Ciekawostka

Istnieje jeszcze inna metoda rozwiązania zadania – wszystkie litery w sprawdzanych ciągach znaków zamienia się na wielkie i wtedy dopiero je porównuje.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Wstaw do programu pętlę, w której do pewnego łańcucha znaków `odwrot` wstawisz lustrzane odbicie łańcucha znaków `tekst` o długości co najmniej dwóch znaków. Swój program przetestuj dla łańcucha znaków *owocowo.*

Specyfikacja problemu:

Dane:

- *tekst* – łańcuch znaków o długości przynajmniej dwóch znaków
- *odwrot* – łańcuch znaków; lustrzane odbicie łańcucha *tekst*

Wynik:

Program wyświetla komunikat *Palindrom* (jeśli tekst jest palindromem) lub *Nie palindrom* (w przeciwnym wypadku).

Ćwiczenie 2

Napisz program, który sprawdzi, czy dane wyrażenie zawarte w pewnym łańcuchu znaków `tekst` (liczącym przynajmniej dwa znaki) jest palindromem. Przetestuj program dla wyrażenia *mamo mam omam*.

Specyfikacja problemu:

Dane:

- *tekst* – łańcuch znaków o długości przynajmniej dwóch znaków

Wynik:

Program wyświetla komunikat *Palindrom* (jeśli tekst jest palindromem) lub *Nie palindrom* (w przeciwnym wypadku).

Ćwiczenie 3

Pewna firma z branży lotniczej miała problem z przekłamaniami transmisji danych – zdarzało się, że urządzenie nadawcze wysyłało bit 1, który jednak był interpretowany przez odbiornik jako 0. Uznano, że rozwiązaniem tego problemu będzie zastosowanie następującego kodu: po każdych ośmiu bitach nadawane są te same bity, ale w odwrotnej kolejności. Dzięki takiemu rozwiązaniu można określić, czy otrzymane dane są poprawne. Napisz program, który określi, czy podany ciąg zer i jedynek jest poprawnym kodem (ma poprawną strukturę oraz poprawną długość). Dla prawidłowych kodów powinien drukować wiadomość *Poprawny kod*, a dla nieprawidłowych: *Niepoprawny kod*. Przetestuj jego działanie dla ciągu bitów *1001110110111001*.

Specyfikacja problemu:

Dane:

- *dane* – łańcuch znaków o długości 16; ciąg bitów składający się z dwóch połączonych ciągów bitów; pierwszy to nadane osiem bitów, a drugi to ciąg tych samych ośmiu bitów w odwróconej kolejności

Wynik:

Program wyświetla informację na temat tego, czy kod *dane* jest poprawny, czy nie. Wyświetla komunikat *Poprawny kod* lub *Niepoprawny kod*.

Dla nauczyciela

Autor: Zespół autorski Contentplus.pl sp. z o.o.

Przedmiot: Informatyka

Temat: Palindromy w języku C++

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:

b) na tekstach: porównywania tekstów, wyszukiwania wzorca w tekście metodą naiwną, szyfrowania tekstu metodą Cezara i przestawieniową,

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Przeanalizujesz napisany w C++ program sprawdzający, czy dany ciąg znaków jest palindromem.
- Porównasz różne sposoby sprawdzania, czy ciąg znaków jest palindromem.
- Zaimplementujesz algorytm tworzenia palindromu.

Strategie nauczania:

- konstruktywizm;
- konektywizm;
- myślenie komputacyjne.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Palindromy w języku C++”. Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. Chętna lub wybrana osoba przypomina najważniejsze informacje dotyczące palindromów.

2. Przedstawienie tematu zajęć oraz wspólne z uczniami ustalenie kryteriów sukcesu.
3. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie analizują treści z sekcji „Przeczytaj" wyświetlone na tablicy.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Symulacja interaktywna". Uczniowie wspólnie zapoznają się z prezentacją, a następnie indywidualnie wykonują ćwiczenie 1. Chętne lub wybrane osoby prezentują kod na forum klasy. Nauczyciel komentuje go i wskazuje ew. błędy.
3. Nauczyciel dzieli uczniów na grupy czteroosobowe. Uczniowie wykonują ćwiczenia nr 2 i 3, a następnie porównują swoje odpowiedzi z inną grupą.

Faza podsumowująca:

1. Uczniowie odpowiadają na pytania postawione na początku zajęć.
2. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.
3. Wybrany uczeń podsumowuje zajęcia z programowania w C++, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie analizują przedstawioną w na schemacie interaktywnym w sekcji „Symulacja interaktywna" metodę sprawdzania, czy dany tekst jest palindromem.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.

- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj", „Symulacja interaktywna", „Sprawdź się" jako materiał do lekcji powtórkowej.
- Nauczyciel może zwrócić uczniom uwagę na temat tego, jakiej minimalnej długości powinien być łańcuch znaków, by móc mówić o palindromie.