Łańcuchy znaków w języku C++

- Wprowadzenie
- Film samouczek
- Przeczytaj
- Sprawdź się
- Dla nauczyciela

Źródło: Danielle MacInnes, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Poznaliśmy już znakowy typ zmiennej (e-materiał Łańcuchy znaków. Do tworzenia słów i analizowania tekstów potrzebujemy jeszcze specjalnej tablicy znaków.

W tym e-materiale omówimy łańcuchy znaków i ich zastosowanie w języku C++. Wiedza ta przyda się nam do rozwiązania zadań maturalnych, które często wymagają przeprowadzenia operacji na łańcuchach znaków.

Implementację łańcuchów znaków w pozostałych językach oprogramowania znajdziesz w e-materiałach:

- Łańcuchy znaków w języku Python,
- Łańcuchy znaków w języku Java.

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

> **Twoje cele**
>
> - Przeanalizujesz zastosowanie łańcuchów znaków w języku C++.
> - Prześledzisz, jak poprawnie używać łańcuchów w języku C++.

- Rozwiążesz, często pojawiające się w zadaniach maturalnych, problemy dotyczące łańcuchów znaków.

Film samouczek

Polecenie 1
Wykonanie wskazanych operacji na ciągu znaków.

Na ciągu znaków `Ala ma kota, a Marysia ma psa` wykonaj operacje:

- sprawdź długość napisu;
- dodaj informację o imieniu psa;
- zmień treść zdania tak, aby informowało, że Marysia ma rybę;
- informacje o Marysi skopiuj do nowej zmiennej;
- zapisz informacje o Marysi wielkimi literami.

Dane:

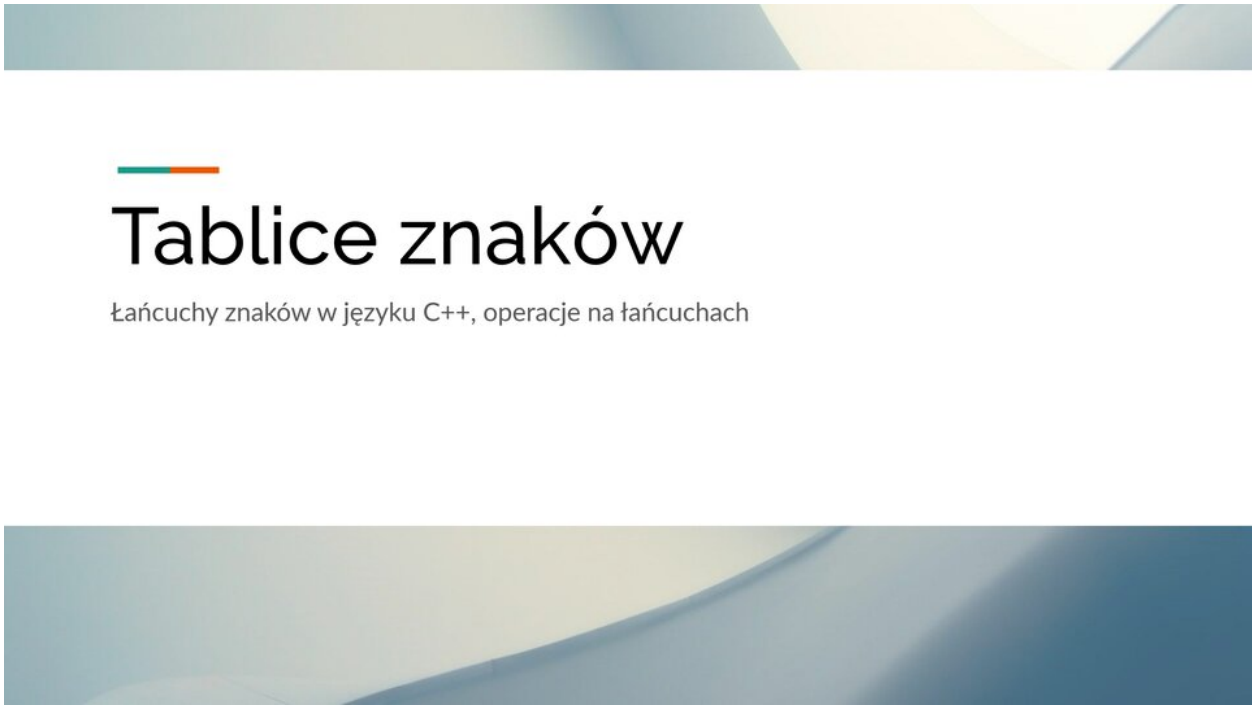
- `zdanie` – łańcuch znaków „Ala ma kota, a Marysia ma psa”
- `Burek` – łańcuch znaków; nowe imię psa
- `rybę` – łańcuch znaków, który ma zastąpić łańcuch znaków `psa`

Wynik:

- `zdanie.length()` – długość łańcucha znaków `zdanie`; liczba naturalna
- `zdanieOMarysi` – nowy łańcuch znaków, który ma przechowywać skopiowane informacje o Marysi.

Polecenie 2
Porównaj swoje rozwiązanie z filmem.

Trwa wczytywanie danych ..



Film dostępny pod adresem /preview/resource/RncRbZZIv4Lkh

Film nawiązujący do treści materiału: Łańcuchy znaków w języku C++, operacje na łańcuchach.

Plik o rozmiarze 831.00 B w języku polskim

Polecenie 3

Przeczytaj

Na łańcuchach znaków operujemy, sprawdzając, czy podane słowa są palindromami lub anagramami. Używamy ich również w niektórych szyfrach, takich jak na przykład szyfr Cezara.

Biblioteka `string`

Aby stworzyć łańcuch `string` oraz skorzystać z funkcji opisanych w tej lekcji, musimy najpierw dołączyć do programu bibliotekę `string`. Są w niej zawarte różne funkcje umożliwiające wykonywanie operacji na łańcuchach znaków.

Bibliotekę `string` dołączymy, wpisując następujący kod:

```
#include <string>
```

Wykorzystamy również przestrzeń nazw:

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

Sposoby pobierania tekstu

Załóżmy, że chcemy pobrać od użytkownika jego imię i nazwisko. Możemy to zrobić, używając predefiniowanego strumienia wejścia `cin`.

Prawdopodobnie początkowo zapiszemy instrukcje w taki sposób:

```
string imieINazwisko;
cout << “Podaj swoje imię i nazwisko: ";
cin >> imieINazwisko;
```

Pozwoli to co prawda przyjąć dane od użytkownika, ale może się zdarzyć, że nie zostaną one wczytane w całości. Jeśli pomiędzy imieniem i nazwiskiem pojawi się odstęp (spacja), pobrane zostanie jedynie imię. Wynika to z faktu, że strumień `cin` traktuje wszystkie białe znaki jako koniec wpisywanego ciągu.

Istnieją dwa rozwiązania tego problemu. Pierwszym z nich byłoby poproszenie użytkownika o wpisanie najpierw imienia, potem nazwiska:

```
string imie;
string nazwisko;
cout << “Podaj swoje imie: ";
cin >> imie;
cout << “Podaj swoje nazwisko: ";
cin >> nazwisko;
```

Możemy również pobrać dane za pomocą funkcji `getline()`. W tym przypadku kod wyglądałby następująco:

```
string imieINazwisko;
cout << “Podaj swoje imię i nazwisko: ";
getline(cin, imieINazwisko);
```

W funkcji `getline()` pierwszym argumentem jest strumień, za pomocą którego chcemy pobrać dane, a drugim `string` – do którego chcemy zapisać ciąg znaków. Ponieważ funkcja `getline()` nie traktuje spacji jako końca strumienia danych, nadaje się do pobierania całych wierszy tekstu, w przypadku których wyrazy rozdzielone są spacjami.

Funkcje `length()` i `size()`

Są to metody, które zwracają informację o długości łańcucha znaków.

Zmienne łańcuchowe typu `string`, w przeciwieństwie do zwykłych tablic, dają nam możliwość sprawdzenia liczby znaków składających się na ciągi. Jest to szczególnie przydatne w sytuacjach, w których np. mamy zliczyć, ile razy dany znak występuje w ciągu. Załóżmy, że chcemy policzyć, ile zer pojawiło się w ciągu zer i jedynek. Zrobimy to w następujący sposób:

```
string ciag = "00001";
int liczbaZer = 0;

for (int i = 0; i < ciag.size(); i++) {
    if (ciag[i] == '0') liczbaZer++;
}
```

Jak widać, metody te mogą posłużyć za wartość, której użyjemy do określenia warunku powtórzenia pętli. Użyliśmy tutaj metody `size()`, ale równie dobrze w jej miejsce moglibyśmy wstawić `length()`. Nie ma między nimi większej różnicy, a efekt zastosowania jest ten sam.

Co prawda w pokazanym przykładzie moglibyśmy z łatwością sami policzyć, z ilu znaków składa się łańcuch – nie jest to duża ani zmieniająca się wartość. W przypadku, gdy łańcuch znaków jest o wiele dłuższy lub jego długość nie jest stała, wykonanie tej samej operacji bez użycia wspomnianych metod byłoby trudne.

W zadaniach maturalnych często wymaga się, by zdający wykonali operacje na dużej liczbie ciągów znaków zapisanych w pliku tekstowym. Oznacza to konieczność wielokrotnego wczytania każdego ciągu do łańcucha znaków o tej samej nazwie. Ponieważ są one różnej długości, metody `length()` oraz `size()`, okazują się niezwykle przydatne.

Zmiana zawartości łańcucha znaków

Każdy znak w łańcuchu znaków ma swój indeks, który jest liczbą naturalną. Numerację zaczynamy od `0`. Dla łańcucha znaków `test` indeks pierwszej litery `t` to `0`, a indeks litery `s` to 2.

Załóżmy, że chcemy znaleźć, a następnie wypisać pozycję ciągu znaków `test` w łańcuchu znaków `To jest wpis testowy.` Służy do tego funkcja `find()`, której nazwę łatwo jest zapamiętać, ponieważ **znajduje** ona (ang. *find*) pozycję podanego ciągu znaków.

```
1 string tekst = “To jest wpis testowy";
2 cout << tekst.find(“test");
```

Gdy znajdziemy już pozycję wspomnianego ciągu znaków, możemy jej użyć w kolejnej funkcji, aby zamienić odszukany wyraz na inny. Skorzystamy przy tym z funkcji `replace()`.

Załóżmy, że w łańcuchu `To jest wpis testowy` chcemy zmienić wyraz `testowy` na `nowy`. Aby to zrobić, wpiszemy następującą linię kodu:

```
1 tekst.replace(tekst.find(“test"), 4, “n");
```

> **Ważne!**
>
> Zauważmy, że literę `n` wpisaliśmy w cudzysłowie, nie zaś między apostrofami tak jak wcześniej. Wynika to z faktu, że trzeci argument funkcji `replace()` jest typu `string`. Apostrofy służą zaś do oznaczania pojedynczego znaku, czyli typu `char`.

Po zakończeniu działania funkcji zawartość łańcucha zmieni się z `To jest wpis testowy` na `To jest wpis nowy`.

Słownik

> **biały znak**
>
> każdy znak, który jest niewidoczny; przykładami białych znaków są spacja, znak tabulacji, znak końca linii lub dowolny inny znak, który nie ma kształtu na ekranie

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Napisz program, który zlokalizuje dany łańcuch znaków w łańcuchu znaków `tekst`, a następnie zamieni go na inny łańcuch znaków.

Działanie programu przetestuj dla zmiennej `tekst = Umiem operowac na zmiennych`, w której łańcuch znaków `zmiennych` zmienisz na `łańcuchach`.

Specyfikacja problemu:

Dane:

- `tekst` – łańcuch znaków

Wynik:

Program wyświetla nowoutworzony łańcuch znaków.

Ćwiczenie 2

Napisz program, który policzy, ile razy dana litera pojawia się w łańcuchu znaków `tekst`.

Przetestuj działanie programu dla łańcucha
znaków `Bo przecież jutro też jest dzień` oraz litery `o`.

Specyfikacja problemu:

Dane:

- `tekst` – łańcuch znaków
- `litera` – łańcuch znaków

Wynik:

- `licznik` – liczba naturalna

Ćwiczenie 3

Napisz program, który odwraca kolejność liter w łańcuchu znaków `slowo` i wyświetla na ekranie łańcuch znaków zapisany wspak.

Przetestuj działanie programu dla słowa `informatyka`.

Specyfikacja problemu:

Dane:

- `slowo` – łańcuch znaków

Wynik: Program wyświetla łańcuch znaków `slowo` zapisany wspak.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Łańcuchy znaków 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:

1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).

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,

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

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

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

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki;

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 zastosowanie łańcuchów znaków w języku C++.
- Prześledzisz, jak poprawnie używać łańcuchów w języku C++.
- Rozwiążesz, często pojawiające się w zadaniach maturalnych, problemy dotyczące łańcuchów znaków.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- metody aktywizujące.

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: „Łańcuchy znaków w języku C++". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Film samouczek".

Faza wstępna:

1. Nauczyciel wyświetla temat oraz cele zajęć, omawiając lub ustalając razem z uczniami kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. Uczniowie w parach rozwiązują zadanie opisane w Poleceniu 1.
2. Chętne lub wybrane osoby prezentują swoje rozwiązanie. Klasa dyskutuje na jego temat. Nauczyciel komentuje kod, wyjaśnia ew. wątpliwości.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1 z sekcji „Sprawdź się", a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.
4. W kolejnym etapie uczniowie dobierają się w pary i wykonują ćwiczenia nr 2 i 3 z sekcji „Sprawdź się". Następnie konsultują swoje rozwiązania z inną parą uczniów i ustalają jedną wersję odpowiedzi.

Faza podsumowująca:

1. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy w zakresie programowania w języku C++.

Praca domowa:

1. Uczniowie rozwiązują Polecenie 2 z sekcji „Film samouczek".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać multimedium w sekcji „Przeczytaj” do przygotowania się do lekcji powtórkowej.