Sortowanie pozycyjne słów w języku C++

Źródło: Markus Spiske, domena publiczna.

Z sortowaniem słów mamy do czynienia w słowniku, encyklopedii czy dzienniku szkolnym. W e-materiale Sortowanie pozycyjne słów dowiedzieliśmy się, jak do porządkowania słów użyć algorytmu sortowania pozycyjnego.

W tym e-materiale zaimplementujemy ten algorytm w języku C++.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych lekcjach z tej serii:

- Sortowanie pozycyjne słów w języku Java,
- Sortowanie pozycyjne słów w języku Python.

Więcej zadań? Sięgnij do: Sortowanie pozycyjne słów – zadania maturalne.

> **Twoje cele**
>
> - Przeanalizujesz algorytm sortujący słowa w porządku leksykograficznym zaimplementowany w języku C++.
> - Prześledzisz symulację algorytmu sortowania pozycyjnego słów.
> - Zaimplementujesz algorytm sortowania pozycyjnego w języku C++.

Przeczytaj

Implementacja algorytmu sortowania pozycyjnego słów w języku C++

Jak już wiesz, sortowanie pozycyjne służy do sortowania zbiorów danych według porządku leksykograficznego. Algorytm ten wewnętrznie używa pomocniczego – innego niż sortowanie pozycyjne – algorytmu sortowania, który jest odpowiedzialny za sortowanie według poszczególnych elementów w ciągach.

Pamiętajmy, że zastosowany algorytm musi być stabilny – możemy użyć dowolnego algorytmu, jeśli tylko spełnia on ten warunek. Sortowanie pozycyjne jest często łączone np. z sortowaniem przez zliczanie, ponieważ oba algorytmy są relatywnie szybkie dla rozważanego typu danych (ciągów znaków).

Na podstawie zdobytej wiedzy przygotujmy program sortujący zbiór słów w porządku leksykograficznym. Poniżej została przedstawiona jego specyfikacja.

Specyfikacja problemu:

Dane:

- `dane[]` – tablica ciągów znaków zawierająca teksty do posortowania, składające się wyłacznie z małych liter alfabetu łacińskiego
- `liczbaElementow` – liczba naturalna; liczba elementów w tablicy `dane`
- `dlugoscNajdluzszegoSlowa` – liczba naturalna; długość najdłuższego słowa w tablicy `dane`

Wynik:

- `dane[]` – posortowana leksykograficznie tablica ciągów znaków; jej elementy wypisane są w kolejnych liniach

Poniżej przedstawimy implementację algorytmu sortowania pozycyjnego w języku C++ i przetestujemy jego działanie dla następujących danych:

- tablicy `dane` zawierającej ciągi znaków: „igor”, „ala”, „adam”
- liczby elementów w tablicy `dane` wynoszącej 3
- najdłuższego słowa o długości 4

Kod zawiera trzy funkcje realizujące algorytm. Zadaniem funkcji `przygotujNapisy()` jest przygotowanie ciągów w tablicy `dane` tak, aby wszystkie miały taką samą długość równą wartości zmiennej `dlugoscNajdluzszegoSlowa`. Jeżeli któryś z ciągów jest krótszy,

zostaje na końcu (po ostatnim znaku) uzupełniony znakami grawisu – ` (jest to ostatni – przed małymi literami – znak w tablicy ASCII). Funkcja `wyczyscNapisy()` ma zadanie odwrotne – po wykonaniu sortowania musi usunąć nadmiarowe znaki grawisów, zastępując je spacjami. Funkcja `sortowaniePrzezZliczanie()` realizuje algorytm sortowania przez zliczanie.

```
#include <iostream>

using namespace std;

string dane[] = { "igor", "ala", "adam" };
const int liczbaElementow = 3;
int dlugoscNajdluzszegoSlowa = 4;

void przygotujNapisy() {
    for (int i = 0; i < liczbaElementow; i++) {
        while (dane[i].length() < dlugoscNajdluzszegoSlowa) {
            dane[i] = dane[i] + "`";
        }
    }
}

void wyczyscNapisy() {
    for (int i = 0; i < liczbaElementow; i++) {
        for (int j = 0; j < dlugoscNajdluzszegoSlowa; ++j) {
            if (dane[i][j] == '`')
                dane[i][j] = ' ';
        }
    }
}


void sortowaniePrzezZliczanie(int indeksZnaku) {
    string temp[liczbaElementow] = {};
    int tablicaZliczenLiter[27] = {};

    for (int i = 0; i < liczbaElementow; i++) {
        int odczytanyKodZnaku = dane[i][indeksZnaku] - 96;
        tablicaZliczenLiter[odczytanyKodZnaku]++;
    }

```

```
    for (int i = 1; i < 27; i++) {
        tablicaZliczenLiter[i] = tablicaZliczenLiter[i] + tablica
    }

    for (int i = liczbaElementow - 1; i >= 0; i--) {
        int odczytanyKodZnaku = dane[i][indeksZnaku] - 96;
        int indeksWTablicyWynikowej = tablicaZliczenLiter[odczyta
        temp[indeksWTablicyWynikowej] = dane[i];
        tablicaZliczenLiter[odczytanyKodZnaku]--;
    }

    for (int i = 0; i < liczbaElementow; i++) {
        dane[i] = temp[i];
    }
}

int main()
{
    przygotujNapisy();

    for (int i = dlugoscNajdluzszegoSlowa - 1; i >= 0; i--) {
        sortowaniePrzezZliczanie(i);
    }

    wyczyscNapisy();

    for (int i = 0; i < liczbaElementow; i++) {
        cout << dane[i] << endl;
    }

   return 0;
}
```

Efektem działania programu jest wyświetlenie danych ciągów znaków już po posortowaniu. Ciągi wyświetlane są w kolejnych wierszach:

```
adam
ala
igor
```

Słownik

porządek leksykograficzny

sposób porządkowania ciągów w sposób analogiczny do porządku alfabetycznego

stabilny algorytm sortowania

algorytm sortowania, który dla dwóch równych elementów w zbiorze danych wejściowych zachowuje ich kolejność

Symulacja interaktywna

Polecenie 1

Zapoznaj się z symulacją, która sortuje słowa metodą pozycyjną. Z uwagi na różne kody ASCII wielkich i małych liter wprowadzane wyrazy zostają zmienione w taki sposób, aby składały się tylko z małych liter. Wprowadź do pola tekstowego pojedynczo imiona osób z twojej klasy, dodając każde do listy. Następnie rozpocznij sortowanie, analizując jego przebieg dla kolejnych pozycji znaków.

Zwróć uwagę na fakt, że w sortowaniu pozycyjnym najważniejszą rolę odgrywa ostatni krok. Istotne jest, aby sortowanie w każdym kroku było stabilne. Dzięki temu zapewniamy poprawną kolejność sortowania w późniejszych krokach, w przypadku, gdy klucz ma tę samą wartość.

Polecenie 2

Przygotuj notatkę ze swoimi spostrzeżeniami dotyczącymi symulacji.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Napisz program sortujący tablicę ciągów znaków zgodnie z porządkiem leksykograficznym, a następnie wypisujący ostatni element posortowanej tablicy. Zwróć uwagę, że elementy tablicy zapisane są wielkimi literami. Działanie swojego programu przetestuj dla następujących danych:

- `dane = {"WODA", "ZUPA", "KAWA", "LODY", "RYBA", "OWOC"}`
- `liczbaElementow = 6`
- `dlugoscNajdluzszegoSlowa = 4`

Specyfikacja:

Dane:

- `dane[]` – tablica ciągów znaków do posortowania; każdy element składa się wyłącznie z dużych liter alfabetu łacińskiego
- `liczbaElementow` – liczba naturalna; liczba elementów w tablicy `dane`
- `dlugoscNajdluzszegoSlowa` – liczba naturalna; długość najdłuższego słowa w tablicy `dane`

Wynik:

- `ostatni` – ciąg znaków, ostatni element posortowanej tablicy `dane`

Twoje zadania

1. Program sortuje tablicę `dane`, a następnie wypisuje ostatni jej element.

```
#include <iostream>

using namespace std;

string dane[] = {"WODA", "ZUPA", "KAWA", "LODY", "RYBA", "OWOC"};
int liczbaElementow = 6;
int dlugoscNajdluzszegoSlowa = 4;

void sortowaniePrzezZliczanie(int indeksZnaku){
    // Tutaj wpisz kod
}

int main()
{
```

```

```

Ćwiczenie 2

Pewien producent herbaty postanowił zaprojektować etykiety w taki sposób, że wszystkie składniki będą wypisane jeden pod drugim w kolejności leksykograficznej, przy czym napis „herbata” będzie wyróżniony. Dział grafiki oczekuje na informację, w której linii znajdzie się wyróżniony napis „herbata”. Napisz program używający sortowania pozycyjnego słów, aby wyznaczyć pozycję (indeks) tego napisu w posortowanej tablicy. Działanie programu przetestuj dla następujących danych:

- `dane = {"lawenda", "mieta", "malina", "pomarancza", "lipa", "rumianek", "bez", "marakuja", "grana`
- `liczbaElementow = 14`
- `dlugoscNajdluzszegoSlowa = 10`

Specyfikacja:

Dane:

- `dane[]` – tablica ciągów znaków; tablica zawierająca składniki herbaty; każdy element składa się wyłącznie z małych liter alfabetu łacińskiego
- `liczbaElementow` – liczba naturalna; liczba elementów w tablicy `dane`
- `dlugoscNajdluzszegoSlowa` – liczba naturalna; długość najdłuższego słowa w tablicy `dane`

Wynik:

- `pozycja` – liczba całkowita; pozycja (indeks) słowa „herbata” w posortowanej tablicy `dane`

Twoje zadania

1. Program sortuje tablicę słów (składników herbat) `dane`, a następnie wypisuje na standardowe wyjście indeks słowa "herbata" w posortowanej tablicy.

```cpp
#include <iostream>

using namespace std;

string dane[] = {"lawenda", "mieta", "malina", "pomarancza", "lipa", "rumianek", "bez",
"marakuja", "granat", "aronia", "hibiskus", "poziomka", "herbata", "papaja"};
int liczbaElementow = 14;
int dlugoscNajdluzszegoSlowa = 10;

void przygotujNapisy() {
    // Tu uzupełnij kod
}

void wyczyscNapisy() {
```

```cpp

```

Ćwiczenie 3

Lokalny klub przygotowuje nabór do juniorskiej drużyny piłkarskiej dla chłopców urodzonych w latach 2011-2013. Dane otrzymane w zgłoszeniach zostały rozdzielone na dwa zbiory: imion i nazwisk chłopców (zapisane małymi literami oraz oddzielone znakiem grawisu) oraz lat ich narodzin. Aby otrzymać pełną informację na temat kandydata, należy zestawić pozycje o tym samym indeksie z obu tablic.

Przykład:

```
string imionaNazwiska[] = {"maciej'kowalski", "adam'nowak"};
int lataNarodzin[] = {2011, 2013};
```

Na podstawie danych z obu tablic możemy przedstawić dwóch kandydatów: Macieja Kowalskiego urodzonego w 2011 roku oraz Adama Nowaka urodzonego w roku 2013.

Wśród otrzymanych zgłoszeń znalazło się kilku kandydatów urodzonych przed rokiem 2011 oraz po roku 2013. Napisz program używający sortowania pozycyjnego słów, aby posortować w kolejności niemalejącej leksykograficznie zbiór imion i nazwisk kandydatów (przy okazji analogicznie zmieniając pozycje lat narodzin w drugiej tablicy), a następnie wyznacz indeksy oraz lata narodzin kandydatów niekwalifikujących się do przeprowadzanego naboru. Swój program przetestuj dla danych podanych w kodzie programu poniżej. Sortowanie powinno w pierwszej kolejności odbywać się na podstawie imienia, a następnie nazwiska.

Specyfikacja:

Dane:

- `imionaNazwiska[]` – tablica ciągów znaków; tablica zawierająca imiona i nazwiska kandydatów; każdy element składa się wyłącznie z małych liter alfabetu łacińskiego i znaku grawisu
- `lataNarodzin[]` – tablica liczb naturalnych; tablica zawierająca lata narodzin kandydatów.
- `liczbaElementow` – liczba naturalna; liczba elementów w tablicach `imionaNazwiska` i `lataNarodzin`
- `dlugoscNajdluzszegoSlowa` – liczba naturalna; długość najdłuższego słowa w tablicach `imionaNazwiska`

Wynik:

- indeksy oraz lata narodzin niekwalifikujących się kandydatów oraz posortowana niemalejąco (w pierwszej kolejności według imienia) pierwotna lista

Przykładowe wyjście:

```
6 2015
7 2010
8 2009
11 2014
adam sikorski
bartosz oko
cezary kot
kacper kowalski
kamil adamski
kamil wojewoda
krystian orzechowski
mateusz frankowski
piotr bramka
robert nowak
tomasz noga
wiktor kowalczyk
```

Twoje zadania

1. Program wypisuje pozycje w posortowanej tablicy kandydatów urodzonych przed 2011 rokiem lub po 2013 roku oraz lata ich narodzin. Następnie wypisuje w kolejnych wierszach kandydatów posortowanych niemalejąco w porządku leksykograficznym, najpierw sortując według imienia, a w drugiej kolejności – nazwiska.

```cpp
#include <iostream>

using namespace std;

string imionaNazwiska[] = {"kamil`wojewoda", "kacper`kowalski", "adam`sikorski",
"wiktor`kowalczyk", "cezary`kot", "tomasz`noga", "kamil`adamski", "piotr`bramka",
"robert`nowak", "mateusz`frankowski", "krystian`orzechowski", "bartosz`oko"};
int lataNarodzin[] = {2011, 2011, 2013, 2014, 2012, 2011, 2013, 2009, 2012, 2010, 2015, 2011};
int liczbaElementow = 12;
int dlugoscNajdluzszegoSlowa = 20;

void przygotujNapisy() {
    // Tu uzupełnij kod
}
```

```cpp

```

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Sortowanie pozycyjne słów w języku C++

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres 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 rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;

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;

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

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

i) struktury dynamiczne: stos, kolejka, lista (do realizacji algorytmu: ONP, symulacji problemu Flawiusza, sortowania leksykograficznego),

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 algorytm sortujący słowa w porządku leksykograficznym zaimplementowany w języku C++.
- Prześledzisz symulację algorytmu sortowania pozycyjnego słów.
- Zaimplementujesz algorytm sortowania pozycyjnego w języku C++.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

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: „Sortowanie pozycyjne słów w języku C++". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć zawarte w sekcji „Wprowadzenie". Następnie wspólnie z uczniami ustala kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie przystępują do cichego czytania tekstu zawartego w sekcji „Przeczytaj", jeśli nauczyciel - na podstawie raportu na platformie - uważa, że przygotowanie uczniów jest wystarczające, może pominąć tę czynność.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Symulacja interaktywna". Uczniowie wspólnie zapoznają się z symulacją interaktywną, która pozwala na wprowadzenie słów do tablicy oraz posortowanie ich metodą sortowania pozycyjnego. Następnie wspólnie analizują sposób działania algorytmu.
3. **Ćwiczenie umiejętności.** Uczniowie, pracując w parach, wykonują ćwiczenie nr 1 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność pisanych kodów, porównuje je i omawia wraz z uczniami. Wskazuje najbardziej efektywne rozwiązanie.
4. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 2 z sekcji „Sprawdź się", a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

Faza podsumowująca:

1. Nauczyciel zadaje pytania podsumowujące, np. :
   – jak nazywamy zbiór, którego każdy element można przypisać do jednej, konkretnej liczby naturalnej?
   – na czym polega sposób sortowania ciągów zwany porządkiem leksykograficznym?
   – jak nazywamy algorytm, który dla dwóch równych elementów w zbiorze danych wejściowych zachowuje ich kolejność?
2. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".

3. Wybrany uczeń podsumowuje zajęcia z programowania w C++, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie wykonują ćwiczenie 3 z sekcji „Sprawdź się”.

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.