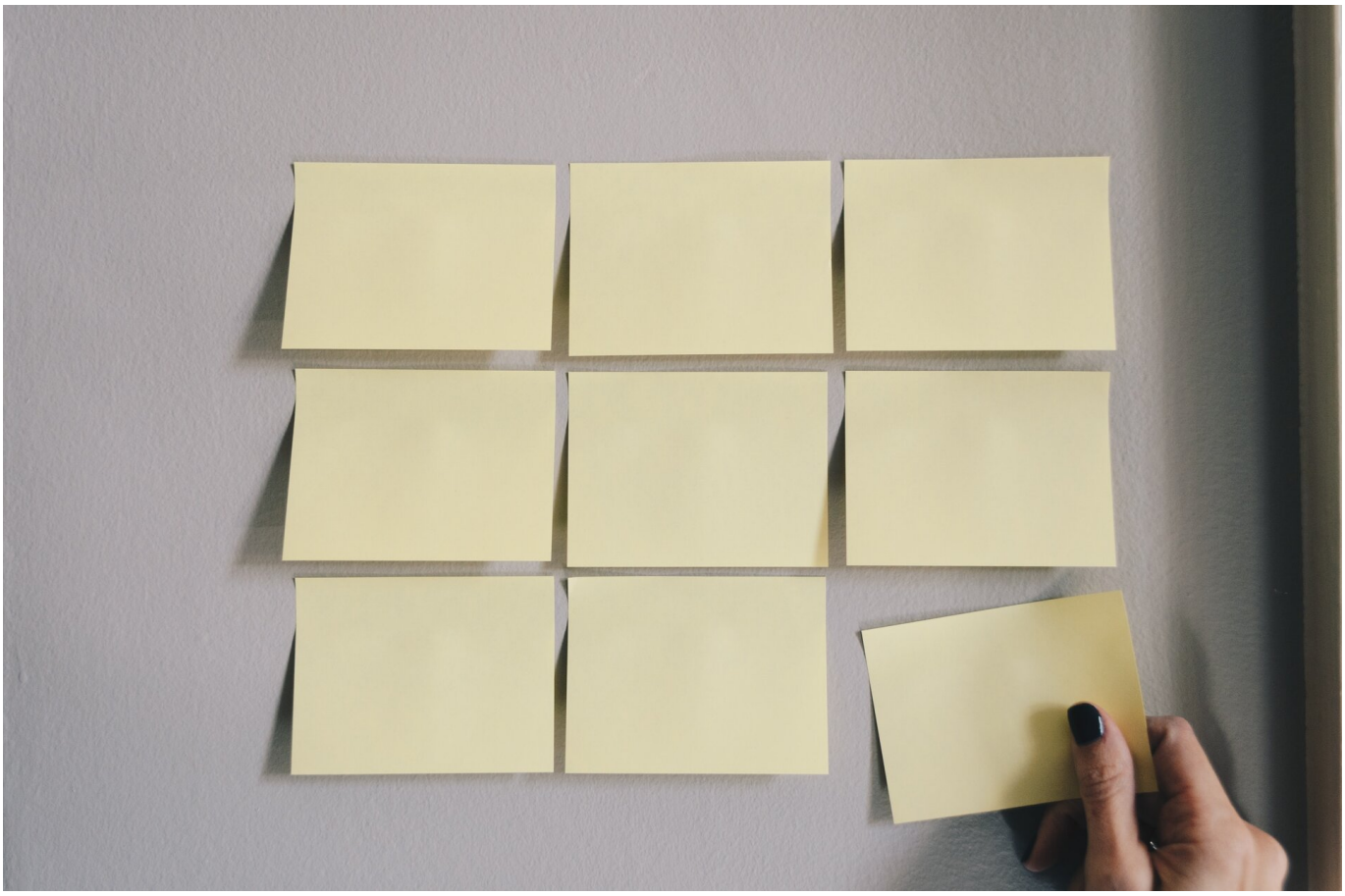
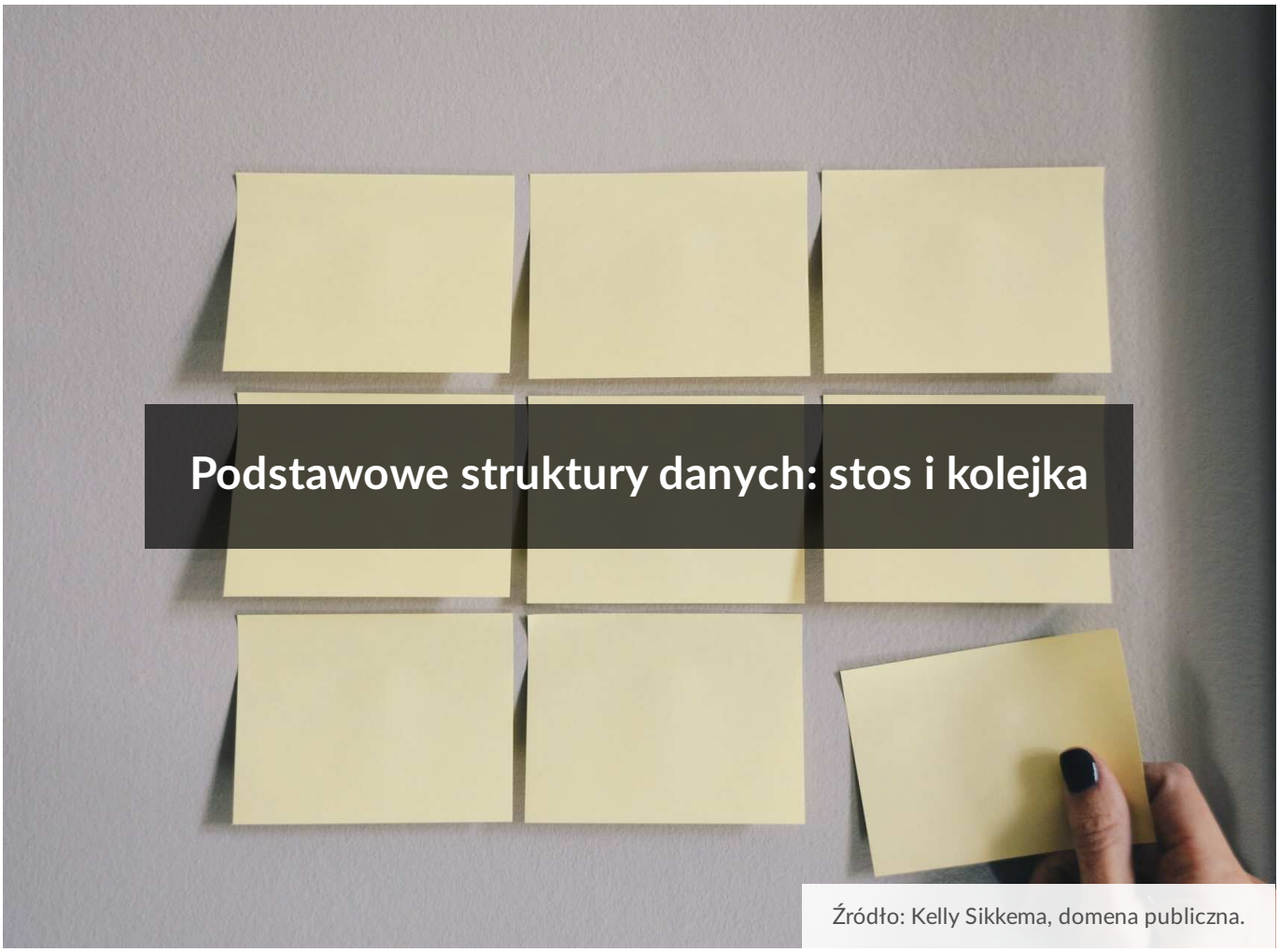


Podstawowe struktury danych: stos i kolejka

- Wprowadzenie
- Przeczytaj
- Aplet
- Sprawdź się
- Dla nauczyciela



Źródło: Kelly Sikkema, domena publiczna.

W pracy programisty często trzeba zastosować odpowiednie struktury danych. Wykorzystując je, możemy rozwiązywać wiele problemów życia codziennego, np. program, w którym zastosujemy kolejkę, usprawni pracę restauracji i sklepu internetowego, czyli miejsc, w których istotna jest kolejność napływających zgłoszeń.

W tym e-materiale omówimy stos i kolejkę, czyli jedne z częściej wykorzystywanych struktur. Więcej informacji o podstawowych strukturach danych znajdziesz w e-materiałach:

- Podstawowe struktury danych,
- Podstawowe struktury danych: tablica,
- Podstawowe struktury danych: rekord,
- Podstawowe struktury danych: lista.

Twoje cele

- Scharakteryzujesz koncepcje stosu i kolejki.
- Prześledzisz terminologię związaną ze stosem i kolejką.
- Przeanalizujesz przykładowe zastosowania stosu i kolejki.

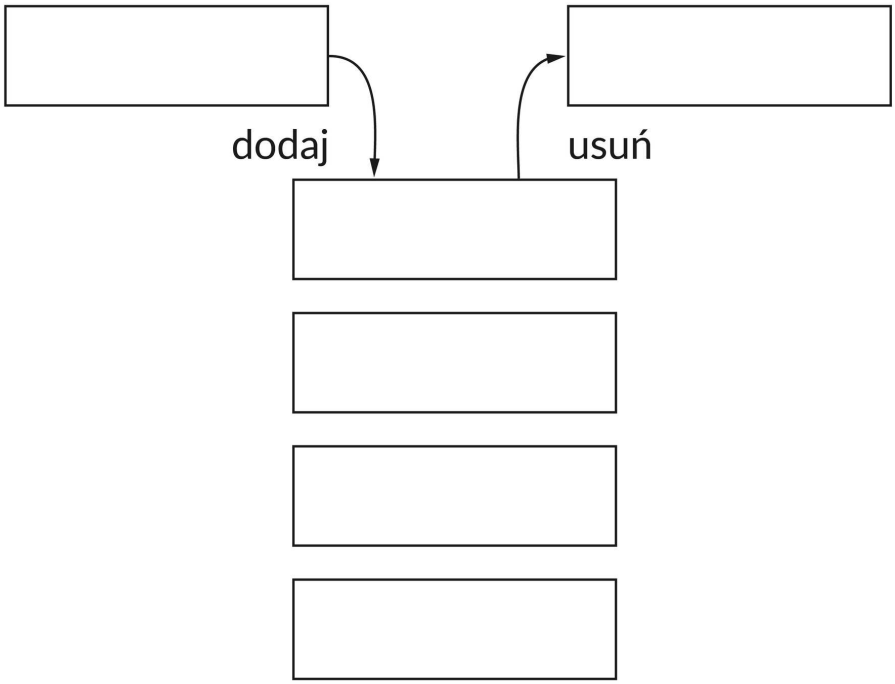
Przeczytaj

Stos

Koncepcja tej struktury dynamicznej jest bardzo prosta: stos przypomina wieżę budowaną z klocków. Nowe elementy układamy na szczycie.

W przypadku stosu mamy dostęp tylko do elementu ostatnio dodanego (umieszczonego najwyżej). Nie mamy do dyspozycji mechanizmu indeksowania, który pozwala odwołać się do pozostałych elementów. Możemy jedynie sprawdzić wartość znajdującą się na samym szczycie.

Podobnie dzieje się w przypadku usuwania elementów ze stosu. Możemy zdjąć z niego tylko element znajdujący się na szczycie.



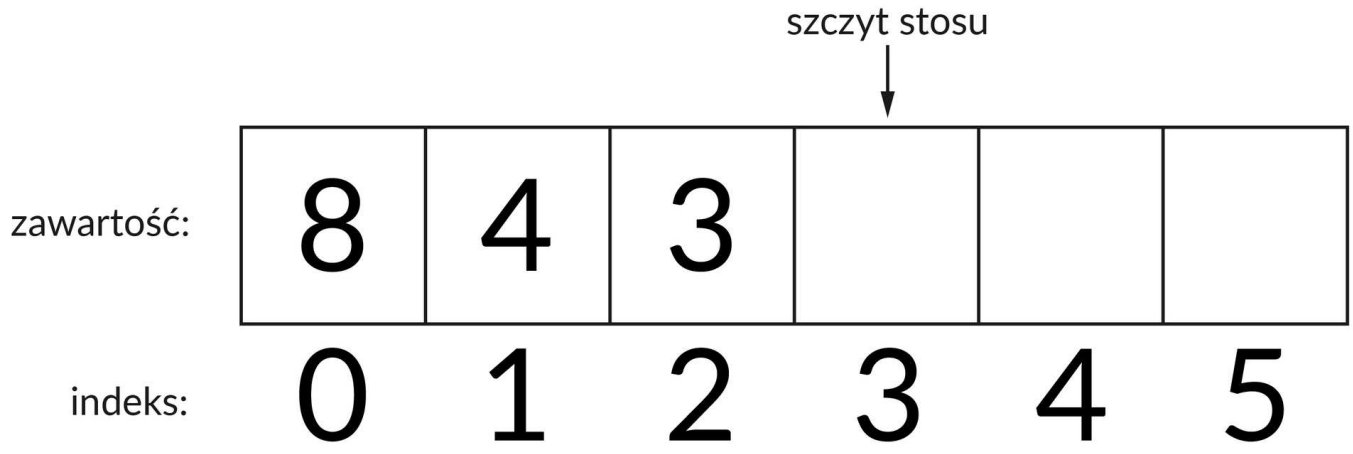
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Bardzo ważną cechą opisanej struktury jest to, że element dodany do niej jako pierwszy, zostanie usunięty jako ostatni. Stos nazywamy strukturą typu **LIFO** (z ang. *Last In First Out*, czyli „ostatni na wejściu, pierwszy na wyjściu”).

Implementacja stosu z zastosowaniem tablicy

Stos jest strukturą dynamiczną – jego rozmiar może się zmieniać w trakcie działania programu. Dla ułatwienia zrozumienia stosu zaimplementujemy go na statycznej tablicy. Musimy jedynie zastanowić się, jaki maksymalny rozmiar powinien mieć budowany stos.

Do poprawnego działania stosu potrzebujemy tylko jednej zmiennej, która będzie przechowywała indeks szczytu stosu. Przy dodawaniu nowego elementu zwiększymy wartość zmiennej o jeden, a przy usuwaniu będziemy ją zmniejszać. Pierwszym („najniższym”) elementem stosu będzie pierwszy element przechowywany w tablicy.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Przeanalizujmy funkcje odpowiedzialne za dodawanie i usuwanie elementów stosu zapisane za pomocą pseudokodu. Zakładamy, że stos jest reprezentowany przez tablicę `stos` o rozmiarze n elementów. Indeks wskazujący szczyt jest przechowywany jako zmienna `indeks_szczytu`. Tablicę indeksujemy od wartości 0.

```
funkcja dodaj_na_stos(a):
    jeżeli indeks_szczytu >= n wykonaj:
        wypisz("Przepełnienie stosu")
    w przeciwnym razie wykonaj:
        stos[indeks_szczytu] ← a
        indeks_szczytu ← indeks_szczytu + 1
```

```
funkcja usun_ze_stosu():
    jeżeli indeks_szczytu = 0 wykonaj:
        wypisz("Stos jest już pusty")
    w przeciwnym razie wykonaj:
        indeks_szczytu ← indeks_szczytu - 1
```

```
funkcja odczytaj_ze_stosu():
    zwróć stos[indeks_szczytu - 1]
```

W zaprezentowanym programie bardzo ważne jest, że elementy nie są faktycznie usuwane z tablicy. Wystarczy jedynie zmniejszyć indeks przechowywany w zmiennej

`indeks_szczytu`, ponieważ przy dodawaniu następnego elementu wartość komórki zostaje nadpisana. Jest to bardzo duże ułatwienie podczas implementowania struktury.

Przykładowe zastosowania stosu

Stos jest wykorzystywany powszechnie w przypadku programów pisanych w językach wysokiego poziomu. Używa się go do przechowywania adresów pamięci i zmiennych, a także jako miejsce do zapisywania zawartości rejestrów procesora.

Implementacja stosu w wybranych językach programowania

Implementację stosu w wybranych językach programowania znajdziesz w następujących e-materiałach:

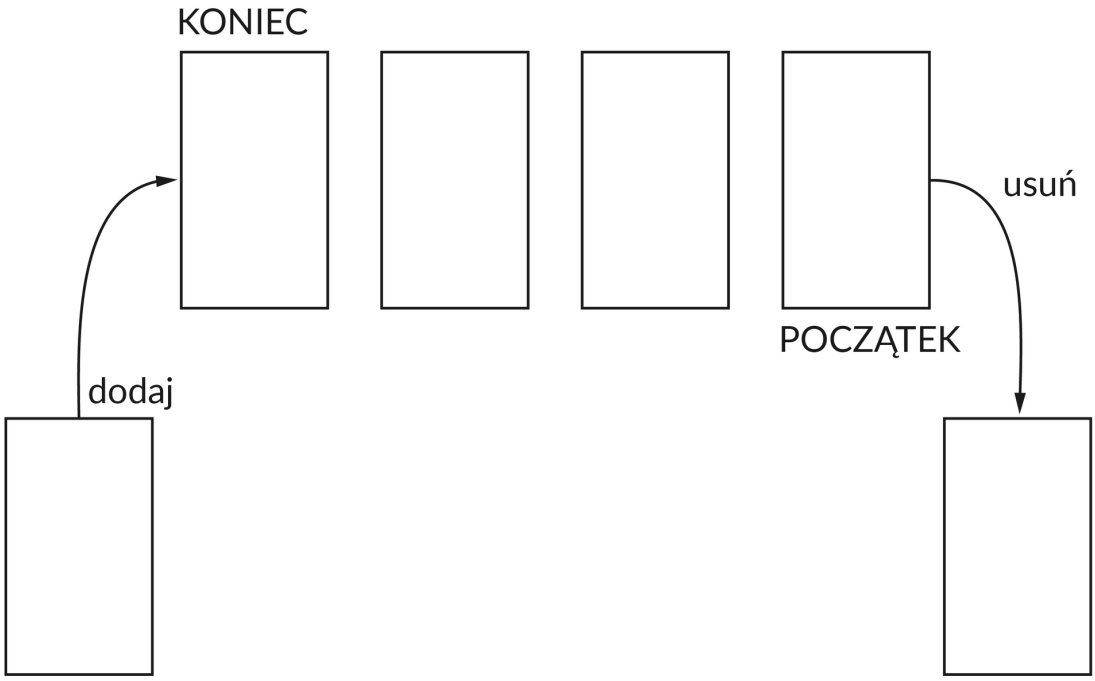
- Dynamiczne struktury danych w języku C++,
- Dynamiczne struktury danych w języku Java,
- Dynamiczne struktury danych w języku Python.

Kolejka

Każdy z nas miał do czynienia z kolejką w sklepie. Zasada jej działania jest dosyć prosta: osoba, która znajduje się przed nami, pojawiła się wcześniej, a więc zostanie wcześniej obsłużona. Tak samo działa struktura danych zwana kolejką.

Kolejka jest przeciwieństwem stosu, jeśli chodzi o sposób dodawania i usuwania elementów. Nazywamy ją strukturą typu **FIFO** (z ang. *First In First Out*, czyli „pierwszy na wejściu, pierwszy na wyjściu”). Nowy element jest dodawany na końcu kolejki. Natomiast jako pierwszy usuwany jest element z początku kolejki.

Kolejka może być wykorzystana np. w teorii grafów w algorytmie przeszukiwania grafu wszerz. Więcej informacji na jej temat znajdziesz w e-materiale Dynamiczne struktury danych.



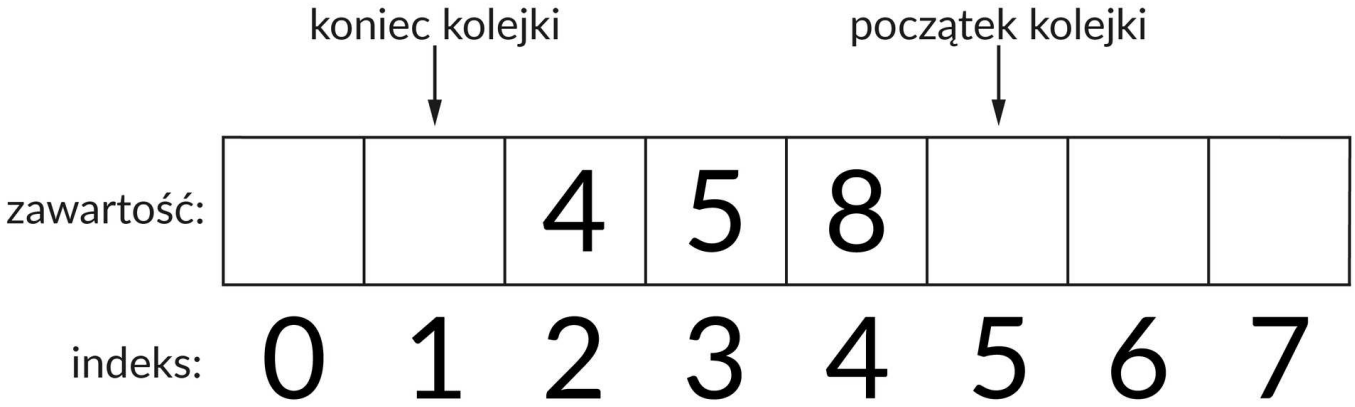
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Tak jak w przypadku stosu, nie możemy użyć indeksowania, aby odwołać się do dowolnych elementów składających się na kolejkę. Jesteśmy w stanie tylko odczytać wartości znajdujące się na jej początku.

Implementacja kolejki z zastosowaniem tablicy

Kolejka także jest strukturą dynamiczną, lecz możemy ją zaimplementować, używając statycznej tablicy. Jedynym ograniczeniem jest maksymalna liczba przechowywanych w kolejce elementów.

Aby kolejka działała poprawnie, konieczne jest użycie dwóch zmiennych. Jedna z nich będzie wskazywała początek kolejki, natomiast druga jej koniec.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Poniżej znajdują się zapisane za pomocą pseudokodu funkcje wykorzystywane podczas dodawania i usuwania elementów kolejki. Zakładamy, że kolejka jest reprezentowana przez tablicę `kolejka` o rozmiarze n elementów. Indeksy początku oraz końca kolejki są

przechowywane w postaci zmiennych `indeks_poczatek` oraz `indeks_koniec`. Początkowa wartość `indeks_poczatek` wynosi $n - 1$. Zmienną `indeks_koniec` inicjalizujemy z kolei wartością o jeden mniejszą, czyli $n - 2$. Tablicę indeksujemy od wartości 0.

```
Funkcja dodaj_do_kolejki(a):
    jeżeli indeks_koniec = indeks_poczatek wykonaj:
        wypisz("Kolejka jest pełna")
    w przeciwnym razie wykonaj:
        kolejka[indeks_koniec] ← a
        indeks_koniec ← indeks_koniec - 1

        jeżeli indeks_koniec < 0:
            indeks_koniec ← n - 1
```

```
Funkcja usun_z_kolejki():
    jeżeli indeks_koniec = (indeks_poczatek - 1)
        LUB indeks_poczatek = 0 ORAZ indeks_koniec = n - 1:
        wypisz("Kolejka jest pusta")
    w przeciwnym razie:
        indeks_poczatek ← indeks_poczatek - 1

        jeżeli indeks_poczatek < 0:
            indeks_poczatek ← n - 1
```

Przedstawiony za pomocą pseudokodu program może wydawać się skomplikowany, ponieważ zawiera dużo instrukcji warunkowych. Odpowiadają one za przesuwanie indeksów wskazujących krańce kolejki z początku na koniec tablicy. Dzięki temu nigdy nie wychodzimy poza zakres tablicy w wyniku modyfikacji indeksów.

Dodatkowo `indeks_poczatek` zawsze wskazuje indeks komórki tablicy za pierwszym elementem. Jest to pewne ułatwienie, które pomaga stwierdzić, kiedy kolejka jest pełna, a kiedy pusta. Jego wadą jest to, że kolejka może przechowywać nie n, lecz $n - 1$ elementów. Aby dokładniej zrozumieć, dlaczego tak się dzieje, pomocne mogą okazać się aplety dostępne w sekcji „Aplet”, których zadaniem jest wizualizowanie działania kolejki.

> **Dla zainteresowanych**
>
> Istnieje również wariant kolejki nazywany **kolejką priorytetową**. Charakteryzuje się ona tym, że elementy znajdujące się w kolejce są zawsze posortowane. W rezultacie na początku kolejki zawsze znajduje się najmniejszy bądź największy element.

Implementacja kolejki w wybranych językach programowania

Implementację kolejki w wybranych językach programowania znajdziesz w następujących e-materiałach:

- Dynamiczne struktury danych w języku C++,
- Dynamiczne struktury danych w języku Java,
- Dynamiczne struktury danych w języku Python.

Słownik

rekurencja

technika programowania, w której funkcja wywołuje samą siebie aż do napotkania przypadku podstawowego

stos

struktura danych, w której informacje są pobierane ze szczytu i na niego odkładane; struktura typu LIFO (*Last In*, *First Out* – ostatni na wejściu, pierwszy na wyjściu)

kolejka

liniowa struktura danych, w której nowe dane dopisywane są na końcu kolejki, a z początku kolejki pobierane są dane do przetwarzania; struktura typu FIFO (*First In*, *First Out*; pierwszy na wejściu, pierwszy na wyjściu)

Aplet

Polecenie 1

Dla restauracji Trattoria Megabajt trzeba napisać program, dzięki któremu klienci będą mogli sprawdzić, czyje zamówienie jest obecnie realizowane. Zanim zaczniemy pisać kod, musimy wybrać strukturę danych służącą do opisania zamówień oraz procesu obsługiwania klientów. Należy to zrobić tak, aby program działał jak najbardziej efektywnie.

Zastanów się nad tym problemem samodzielnie przez pięć minut, a następnie sprawdź, czy wybrana przez ciebie struktura jest odpowiednia.

1

Zastanówmy się najpierw, czy wystarczą w tym przypadku standardowe kontenery na dane?

Do wyboru mamy zmienne oraz tablice.

2

Oczywiście proste typy zmiennych odrzucamy, ponieważ ich dynamiczne generowanie byłoby niepotrzebną komplikacją.

Teoretycznie moglibyśmy użyć tablic. Jednak na dłuższą metę byłoby to rozwiązanie nieefektywne ze względu na ich ograniczenia (tablica musi mieć określony rozmiar).

3

Czy skorzystać ze stosu?

Dostęp do danych na stosie jest szybki, jednak po jego zastosowaniu klienci, którzy złożyli zamówienie jako pierwsi, zostaliby obsłużeni jako ostatni. Tak przecież być nie może.

4

Może użyć listy?

Rzeczywiście, lista mogłaby się w tym przypadku sprawdzić. Jednak okazuje się, że istnieje inna struktura, która jeszcze lepiej rozwiązuje problem.

5

Użyjemy oczywiście kolejki. Działa ona szybko oraz pozwala na obsługiwanie zamówień w takiej kolejności, w jakiej zostały złożone.

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Polecenie 2

Oto inny przykład. Załóżmy, że piszemy aplikację służącą do zarządzania osobistą biblioteką. Zależy nam na tym, aby program pokazywał, które książki zaczęliśmy ostatnio czytać, abyśmy mogli szybko do nich wrócić i dokończyć lekturę. W jakim pojemniku na dane najlepiej będzie przechowywać informacje o ostatnio czytanych książkach?

Zastanów się nad tym problemem samodzielnie przez pięć minut, a następnie sprawdź, czy wybrana przez ciebie struktura jest odpowiednia.

1

Czy standardowe zmienne i tablice się nadają?

W tym przypadku również nie. Mają one zbyt wiele ograniczeń.

2

Czy zastosować kolejkę?

Problem z kolejką polega na tym, że książka, którą zaczęliśmy czytać najwcześniej, będzie zawsze na pierwszym miejscu.

Takie rozwiązanie mogłoby być oczywiście zastosowane, ale nie o taki mechanizm chodzi.

3

Może należy użyć drzewa binarnego?

Niestety, nie chcemy przechowywać książek hierarchicznie, ponieważ nie ma takiej potrzeby.

4

Czy zastosować listę?

Lista sprawdziłaby się jako opis wszystkich książek w bibliotece, ale w naszym przypadku nie wydaje się najkorzystniejsza.

5

Oczywistym wyborem jest stos, ponieważ każda książka, którą zaczniemy czytać, będzie zawsze na pierwszym miejscu. Do starszych książek powrócimy, kiedy przeczytamy te z samego wierzchołka stosu.

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Polecenie 3

Uruchom aplet, aby dokładniej przeanalizować działanie stosu zaimplementowanego z użyciem tablicy. Zwróć uwagę na ograniczenia związane z tego rodzaju implementacją struktury stosu.

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Polecenie 4

Kolejny aplet symuluje działanie kolejki zaimplementowanej przy użyciu tablicy. Sprawdź działanie każdej z metod oraz zastanów się, jakie ograniczenia niesie ze sobą ten sposób implementacji struktury kolejki.

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Wyjaśnij, na czym polega kolejka priorytetowa.

Ćwiczenie 2

Zaznacz poprawną odpowiedź, oceniając prawdziwość stwierdzenia: „do każdego elementu stosu możemy odwołać się dzięki indeksowaniu”.

- Prawda.
- Fałsz.

Ćwiczenie 3

Zaznacz poprawną odpowiedź, oceniając prawdziwość stwierdzenia: „zarówno stos, jak i kolejkę jesteśmy w stanie zaimplementować, korzystając ze statycznej tablicy”.

- Fałsz.
- Prawda.

Ćwiczenie 4

Wybierz zdanie prawdziwe.

- Stos jest strukturą dynamiczną, natomiast kolejka jest strukturą statyczną.
- Zarówno stos, jak i kolejka są strukturami statycznymi.
- Zarówno stos, jak i kolejka są strukturami dynamicznymi.
- Stos jest strukturą statyczną, natomiast kolejka jest strukturą dynamiczną.

Ćwiczenie 5

Zaznacz poprawną odpowiedź, oceniając prawdziwość stwierdzenia: „stos jest strukturą typu FIFO”.

- ◯ Fałsz.
- ◯ Prawda.

Ćwiczenie 6

Załóżmy, że stos jest reprezentowany przez tablicę `stos` o rozmiarze 25 elementów. Indeks jego szczytu jest przechowywany w zmiennej `indeks_szczytu`. Tablicę indeksujemy od 0. Uzupełnij zapisaną za pomocą pseudokodu funkcję dodającą nowy element na stos.

```
Funkcja dodaj_na_stos(a):
    Jeżeli indeks_szczytu >= [          ]:
        wypisz([          ])
    W przeciwnym razie:
        [          ]
        indeks_szczytu ← indeks_szczytu + 1
```

`"Stos jest już pusty"` | `26` | `25` | `stos[indeks_szczytu] ← a`

`"Przepełnienie stosu"` | `stos[a] ← indeks_szczytu` | `50`

Ćwiczenie 7

Zaznacz wszystkie niezbędne elementy potrzebne do zaimplementowania statycznej kolejki.

- ☐ Dynamiczna tablica reprezentująca kolejkę.
- ☐ Zmienna przechowująca indeks końca kolejki.
- ☐ Zmienna przechowująca informację o liczbie elementów znajdujących się w kolejce.
- ☐ Zmienna przechowująca indeks początku kolejki.
- ☐ Statyczna tablica reprezentująca kolejkę.

Ćwiczenie 8

Uzupełnij tekst w taki sposób, aby jego treść była prawdziwa.

Kolejka jest strukturą _____, co oznacza, że jej rozmiar _____ się zmieniać w trakcie działania programu. Nowe elementy są dodawane na jej _____. Usuwanie elementów możliwe jest tylko z jej _____. Nazywamy ją strukturą typu _____.

początek | FIFO | LIFO | koniec | początku | końca | statyczną | nie może | może | dynamiczną

Dla nauczyciela

Autor: Bartosz Zadrożny
Przedmiot: Informatyka

Temat: Podstawowe struktury danych: stos i kolejka

Grupa docelowa:

Szkoła ponadpodstawowa, 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:

1) w zależności od problemu rozwiązuje go, stosując metodę wstępującą lub zstępującą;

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

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

1) projektuje i tworzy rozbudowane programy w procesie rozwiązywania problemów, wykorzystuje w programach dobrane do algorytmów struktury danych, w tym struktury dynamiczne i korzysta z dostępnych bibliotek dla tych struktur;

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):

- Scharakteryzujesz koncepcje stosu i kolejki.
- Prześledzisz terminologię związaną ze stosem i kolejką.
- Przeanalizujesz przykładowe zastosowania stosu i kolejki.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

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.

Przebieg lekcji

Przed lekcją:

1. Uczniowie powtarzają informacje na temat poznanych wcześniej struktur danych.
2. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Podstawowe struktury danych: stos i kolejka". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Metodą burzy mózgów uczniowie podają, jakiego typu dane mogą być przechowywane za pomocą stosu oraz kolejki.
2. Chętna lub wybrana osoba referuje najważniejsze informacje na temat poznanych wcześniej struktur danych.
3. Nauczyciel wprowadza uczniów szczegółowo w temat lekcji i jej cele. Może posłużyć się wyświetloną na tablicy zawartością sekcji „Wprowadzenie".

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj".
2. Podział klasy na 4 grupy. Każdy zespół wykonuje przydzielone zadania: rozwiązuje jedno z poleceń z sekcji „Aplet".
   Reprezentanci grup omawiają rozwiązania na forum klasy. Pozostali uczniowie mogą zadawać im pytania lub uzupełniać informacje.
3. Uczniowie wykonują ćwiczenia 1–6 z sekcji „Sprawdź się".

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat lekcji zawarty w sekcji „Wprowadzenie" i inicjuje krótką rozmowę na temat zrealizowanych celów (czego uczniowie się nauczyli).

Praca domowa:

1. Uczniowie wykonują ćwiczenia 7 i 8 z sekcji „Sprawdź się".
2. Uczniowie proponują inne niż na lekcji przykłady wykorzystania wszystkich poznanych struktur danych.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj", „Aplet", „Sprawdź się" jako materiał do lekcji powtórkowej.