Programowanie strukturalne - wykład 11

dr Piotr Jastrzębski

Struktury danych

Struktury danych

Struktura danych (ang. data structure) – sposób przechowywania danych w pamięci komputera. Na strukturach danych operują algorytmy.

Podczas implementacji programu programista często staje przed wyborem między różnymi strukturami danych, aby uzyskać pożądany efekt. Odpowiedni wybór może zmniejszyć złożoność obliczeniową, ale z drugiej strony trudność implementacji danej struktury może stanowić istotną przeszkodę.

Przykładowe struktury danych to:

- rekord lub struktura (ang. record, struct), logiczny odpowiednik to krotka
- tablica
- lista
- stos
- kolejka
- drzewo i jego liczne odmiany (np. drzewo binarne)
- graf

Abstrakcyjny typ danych

Abstrakcyjny typ danych

Abstrakcyjny typ danych (ang. abstract data type, ADT) – tworzenie i opisywanie w formalny sposób typów danych tak, że opisywane są jedynie własności danych i operacji wykonywanych na nich (a nie przez reprezentację danych i implementację operacji).

Specyfikacja ADT powinna:

- być jednoznaczna i dokładna;
- zawierać wszystkie przypadki warte rozważenia;
- nie zawierać niepotrzebnych informacji.
- Podając specyfikację ADT (dowolnego typu), powinniśmy uwzględnić:
  - nazwę tego typu;
  - dziedzinę;
  - zbiór funkcji;
  - aksjomaty;
  - warunki początkowe.

Przegląd

Kontener

Nazwa typu: Kontener

Własności typu:

- **dostęp**, czyli sposób uzyskiwania dostępu do obiektów kontenera. W przypadku tablic dostęp odbywa się za pomocą indeksu. W przypadku stosów dostęp odbywa się zgodnie z kolejnością LIFO, a w przypadku kolejek - zgodnie z kolejnością FIFO.
- **magazynowanie**, czyli sposób przechowywania przedmiotów pojemnika,
- **translacji**, to jest sposób przechodzenia przedmiotów pojemnika.

Dostępne działania:

- Inicjalizacja kontenera.
- Dodawanie pozycji do kontenera.
- Usuwanie pozycji z kontenera.
- Opróżnienia kontenera.
- Dostęp do pozycji z kontenera.
- Dostęp do liczby pozycji w kontenerze.

Zbiór

Nazwa typu: Zbiór

Właściwości typu:

- matematyczny odpowiednik pojęcia zbioru.
- brak duplikatów.
- brak uporządkowania elementów.

Dostępne działania:

- Inicjalizacja zbioru.
- Suma zbiorów.
- Część wspólna zbiorów.
- Różnica zbiorów.
- Sprawdzenie czy jeden zbiór zawiera się w się w drugim (jest podzbiorem).
- Sprawdzenie, czy pozycja należy do zbioru.
- Sprawdzenie, czy zbiór jest pusty czy nie.
- Dostęp do liczby pozycji w zbiorze.

Opcjonalnie:

- Dodawanie pozycji do zbioru.
- Usuwanie pozycji ze zbioru.
- Iterowanie/enumerowanie zbioru.
- Dostępność do pojemności zbioru (maksymalnej liczby pozycji mogących być dodanych do zbioru).

Prosta lista

Nazwa typu: Prosta lista

Własności typu: Potrafi przechować ciąg pozycji.

Dostępne działania:

- Inicjalizacja listy.
- Określenie, czy lista jest pusta.
- Określenie, czy lista jest pełna.
- Określenie liczby pozycji w liście.
- Dodanie pozycji na końcu listy.
- Przejście przez listę i przetwarzanie każdej pozycji.
- Opróżnienie listy.

Kolejka

Nazwa typu: Kolejka

Własności typu: Potrafi przechować uporządkowany ciąg pozycji.

Dostępne działania:

- Inicjalizacja kolejki.
- Określenie, czy kolejka jest pusta.
- Określenie, czy kolejka jest pełna.
- Określenie liczby pozycji w kolejce.
- Dodanie pozycji z tyłu kolejki.
- Pobranie i usunięcie pozycji z przodu kolejki.
- Opróżnienie kolejki.

Drzewo binarne

Nazwa typu: Drzewo binarne

Własności typu:

- Drzewo binarne jest albo pustym zbiorem węzłów, albo zbiorem węzłów, z których jeden jest oznaczony jako korzeń.
- Z każdego węzła wywodzą się dokładnie dwa drzewa, zwane lewym poddrzewem i prawym poddrzewem.
- Każde poddrzewo jest samo drzewem binarnym (pełnym lub pustym).
- Każdy węzeł zawiera pozycję, przy czym wszystkie pozycje w lewym poddrzewie poprzedzają pozycję zapisaną w korzeniu, a pozycja w korzeniu poprzedza wszystkie pozycje w prawym poddrzewie.

Dostępne działania:

- Inicjalizacja drzewa.
- Określenie, czy drzewo jest puste.
- Określenie, czy drzewo jest pełne.
- Określenie liczby pozycji w drzewie.
- Dodanie pozycji do drzewa.
- Usunięcie pozycji z drzewa.
- Poszukiwanie pozycji w drzewie.
- Przejście po wszystkich pozycjach w drzewie.
- Opróżnianie drzewa.

Krotka

Krotka (ang. tuple) – struktura danych będąca odzwierciedleniem matematycznej n-ki, tj. uporządkowanego ciągu wartości. Krotki przechowują stałe wartości o różnych typach danych – nie można zmodyfikować żadnego elementu, odczyt natomiast wymaga podania indeksu liczbowego żądanego elementu.

Multizbiór

Multizbiór (także wielozbiór, ang. multiset) – uogólnienie pojęcia zbioru, w którym w odróżnieniu od klasycznych zbiorów jeden element może występować wiele razy. Nie jest jednak dana żadna ich kolejność i tym multizbiór różni się od krotki.

Zbiory $\{1, 2, 3\}, \{3, 2, 1\}$ i $\{1, 2, 2, 3\}$ są identyczne.

Multizbiory $\{1, 2, 3\}, \{1, 2, 3\}$ i $\{3, 2, 1\}$ są identyczne, $\{1, 2, 2, 3\}$ jest jednak inny.

Inne

Kolejka priorytetowa - https://pl.wikipedia.org/wiki/Kolejka_priorytetowa

Kopiec - https://pl.wikipedia.org/wiki/Kopiec_(informatyka)

Rekord - https://pl.wikipedia.org/wiki/Rekord_(informatyka)

Tablica mieszająca - https://pl.wikipedia.org/wiki/Tablica_mieszaj%C4%85ca

Tablica asocjacyjna - https://pl.wikipedia.org/wiki/Tablica_asocjacyjna

Lista

Lista

https://www.geeksforgeeks.org/data-structures/linked-list/

Lista jednokierunkowa

```
struct element
{
    int i;
    struct element * next;
};
```

Warianty - z głową (z wartownikiem) i bez.

Lista dwukierunkowa

```
struct element
{
    int i;
    struct element * prev;
    struct element * next;
};
```

Warianty: * z głową * z ogonem * mieszane

Lista cykliczna

```
struct element
{
    int i;
    struct element * next;
};
```

Kolejka

https://www.geeksforgeeks.org/queue-data-structure/

Stos

https://www.geeksforgeeks.org/stack-data-structure/

Inne

https://www.geeksforgeeks.org/data-structures/

Bibliografia

- Stephen Prata, Język C. Szkoła programowania. Wydanie VI, Wyd. Helion, 2016.
- Wikipedia, dostęp online 15.06.2020.