Wykład 3

- **Złożoność i realizowalność algorytmów**
- **Elementarne struktury danych: *stosy, kolejki, listy***

Dynamiczne struktury danych

Lista jest to liniowo uporządkowany zbiór elementów, z których dowolny element można usunąć oraz dodać w dowolnym miejscu.

Pierwszy i ostatni element listy nazywamy *końcami listy*.

Szczególnym przypadkiem listy jest:

stos
(pobrać, odczytać i wstawić element można tylko na końcu listy),

kolejka
(pobrać i odczytać element można tylko na początku listy, a dodać na końcu).

Listy mogą być posortowane
(najmniejszy element jest w korzeniu).

Lista jednokierunkowa posortowana

Aby dodać element do listy posortowanej należy sprawdzić, w którym miejscu powinien się znajdować.

Sprawdzamy od korzenia, schodząc w dół, jeśli element, który chcemy dodać jest większy od badanego węzła i mniejszy od jego następnika, to należy umieścić go między nimi.

Wskaźnik aktualnego węzła ustawia się na dodawany element, a wskaźnik tego elementu na następnik.

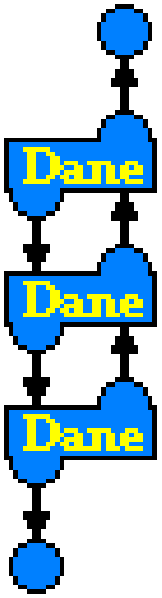
Ponieważ jest to lista jednokierunkowa, przeszukiwanie jej należy zawsze zaczynać od korzenia.

Dodając pierwszy element do pustej listy należy zapamiętać jego wskaźnik, by później móc się do niego przenieść.

W przeciwieństwie do stosu i kolejki listy mogą zawierać dwa wskaźniki.

Listy *dwukierunkowe*

Oto schemat listy dwukierunkowej:



Kolejka jest strukturą liniowo uporządkowanych danych, w której dołączać nowe dane można jedynie na koniec kolejki a usuwać z początku.

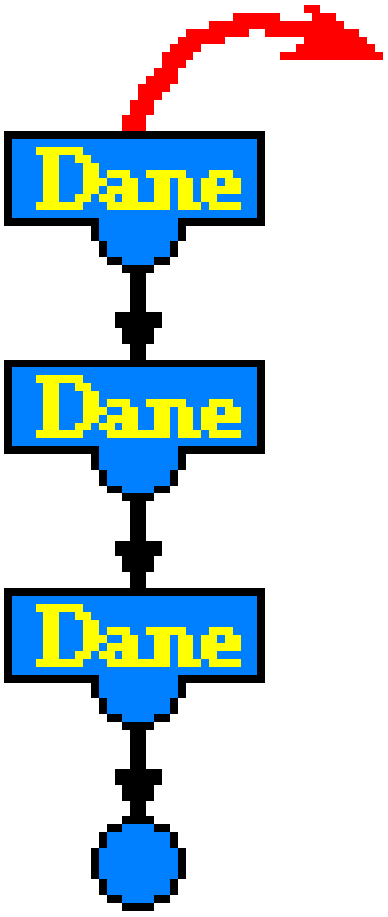
Procedura usunięcia danych z końca kolejki jest taka sama, jak w przypadku **stosu**, z tą różnicą, że usuwamy dane od początku a nie od końca.

Pierwszy element (a dokładniej wskaźnik do jego miejsca w pamięci) musi zostać zapamiętany, by możliwe było usuwanie pierwszego elementu w czasie stałym O(1).

Gdybyśmy tego nie zrobili, aby dotrzeć do pierwszego elementu należałoby przejść wszystkie od elementu aktualnego (czyli ostatniego), co wymaga czasu O(n).

Działanie na kolejce jest intuicyjnie jasne, gdy skojarzymy ją z kolejką ludzi np. w sklepie. Każdy nowy klient staje na jej końcu, obsługa odbywa się jedynie na początku.

Schemat kolejki wygląda następująco:



Stos jest strukturą liniowo uporządkowanych danych, z których jedynie ostatni element, zwany *wierzchołkiem*, jest w danym momencie dostępny.

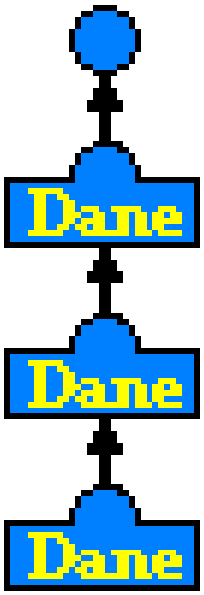
W wierzchołku odbywa się dołączanie nowych elementów, również jedynie wierzchołek można usunąć.

Działanie na stosie jest często porównywane do stosu talerzy:

- nie można usunąć talerza znajdującego się na dnie stosu nie usuwając wcześniej wszystkich innych.

- nie można także dodać nowego talerza gdzieś indziej, niż na samą górę.

Oto schemat stosu:



Drzewo jest bardziej skomplikowaną strukturą niż poprzednie.

Dla każdego drzewa wyróżniony jest jeden, charakterystyczny element- *korzeń*.

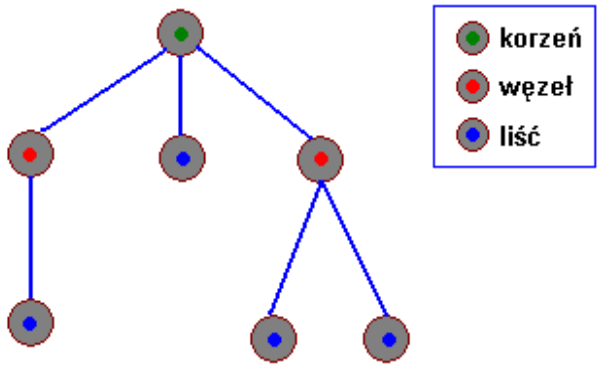
Korzeń jest jedynym elementem drzewa, który nie posiada elementów poprzednich.

Dla każdego innego elementu określony jest dokładnie jeden element poprzedni.

Dla każdego elementu oprócz ostatnich, tzw. *liści* istnieje co najmniej 1 element następny.

Jeżeli liczba następnych elementów wynosi dokładnie 2 to drzewo nazywamy *binarnym*.

Drzewo można zdefiniować, jako acykliczny **graf**.



U

Dla każdego drzewa można określić:

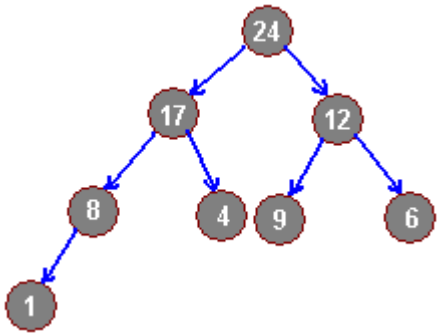
- długość drogi *u* (głębokość) - liczba wierzchołków, przez które należy przejść od korzenia do wierzchołka *u*
- wysokość *u* - maksymalna liczba wierzchołków na drodze od *u* do pewnego liścia
- wysokość drzewa = głębokość = wysokość korzenia +1
- ścieżka z *u* do *v* - zbiór wierzchołków, przez które należy przejść z wierzchołka *u* do *v*
- droga - ścieżka skierowana
- stopień wierzchołka - liczba jego bezpośrednich następników
- stopień drzewa - maksymalny stopień wierzchołka

Kopiec inaczej zwany stogiem jest szczególnym przypadkiem **drzewa binarnego**, które spełnia tzw.*warunek kopca* tzn. każdy następnik jest niewiększy od swego poprzednika.

Z warunku tego wynikają szczególne własności kopca:

- w **korzeniu** kopca znajduje się największy element,
- na ścieżkach (połączeniach między węzłami), od korzenia do liścia, elementy są posortowane nierosnąco

Oto przykładowy kopiec:



Szczególne własności kopców zostały wykorzystane do stworzenia algorytmy do sortowania zwanego **HeapSort**.