STRUKTURY DANYCH

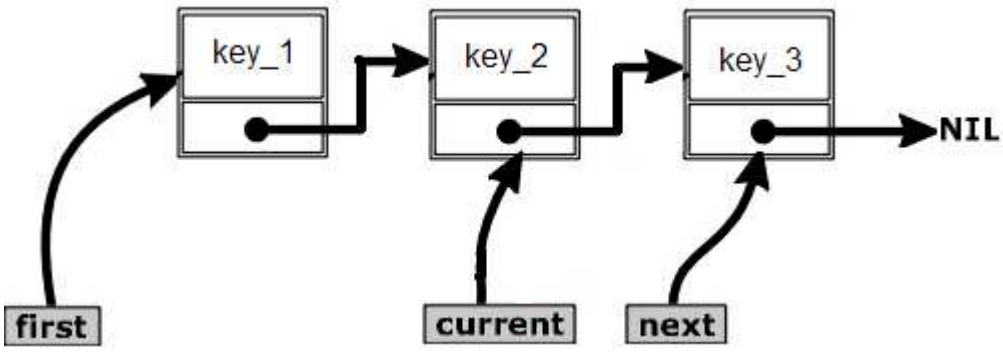
STRUKTURY DANYCH

Wyróżniamy następujące dynamiczne struktury danych:

LISTA

Lista jest liniowo uporządkowaną strukturą zawierającą zbiór elementów, z których dowolny element można usunąć oraz dodać w dowolnym miejscu. Pierwszy i ostatni element listy nazywamy *końcami listy*. Każdy element ma co najwyżej jednego następnika, który składa się z następujących składowych: klucza identyfikacyjnego, wskaźnika do następnika oraz dodatkowo dalszych informacji. Listy mogą być posortowane w porządku niemalejącym (w korzeniu znajduje się element o najmniejszej wartości klucza) lub nierosnącym (w korzeniu znajduje się element o największej wartości klucza). Rozważmy przypadek **jednokierunkowej listy posortowanej** w porządku niemalejącym. Aby dodać nowy element należy sprawdzić, w którym miejscu powinien się on znajdować. Dokonywane jest odpowiednie sprawdzenie rozpoczynające się w korzeniu i przechodzenie przez kolejne elementy w liście, tak długo dopóki nowy element, który chcemy dodać jest większy od badanego węzła i mniejszy od jego następnika, wtedy należy umieścić go między nimi. Należy, więc ustawić wskaźnik aktualnego węzła na dodawany element, a wskaźnik tego elementu na następnik. Ponieważ jest to lista jednokierunkowa, przeszukiwanie jej należy zawsze zaczynać od korzenia. Dodając, więc pierwszy element do pustej listy należy zapamiętać jego wskaźnik, by później mieć dostęp do tej struktury dzięki niemu. Listy mogą zawierać dwa wskaźniki. Takie listy nazywamy **dwukierunkowymi,** które pozwalają na poruszanie się w niej w obydwu kierunkach, co przyspiesza wszystkie operacje.

- *Jednokierunkowa*

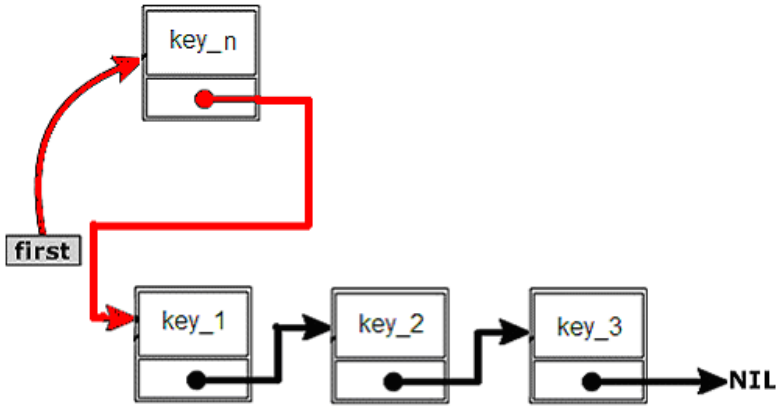


Przeszukiwanie dokonuje się od pierwszego elementu na liście (wskazywanego przez wskaźnik *first*) do ostatniego, który w polu swojego następnika wskazuje na wartość pustą (*NIL*).

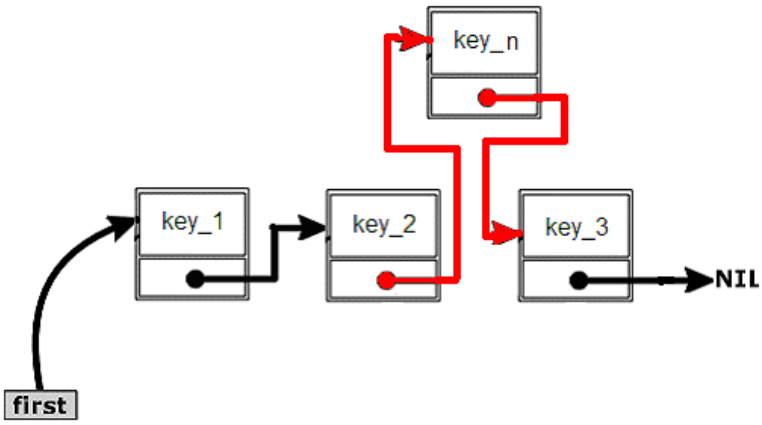
DODAWANIE NOWYCH ELEMENTÓW

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

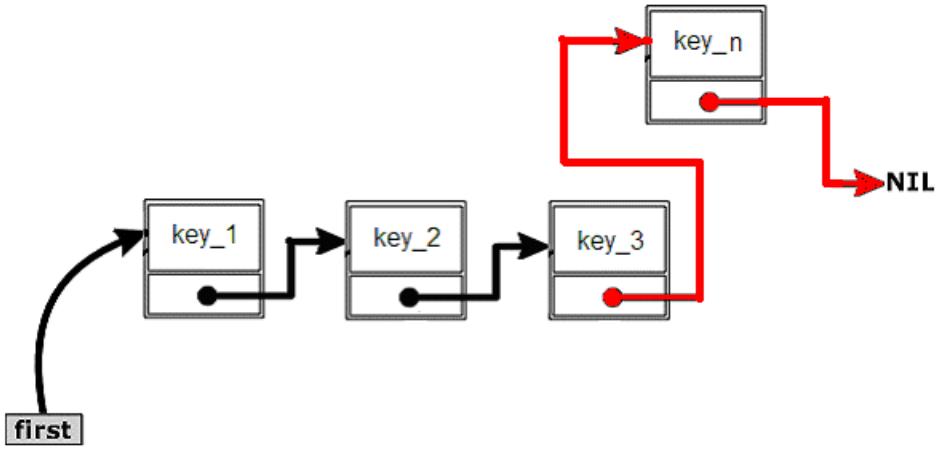
a) na początku



b) w środku (między 2 a 3 elementem)



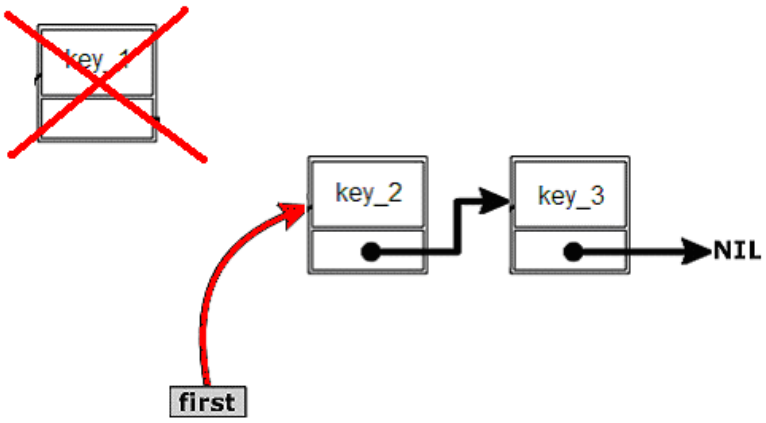
c) na końcu



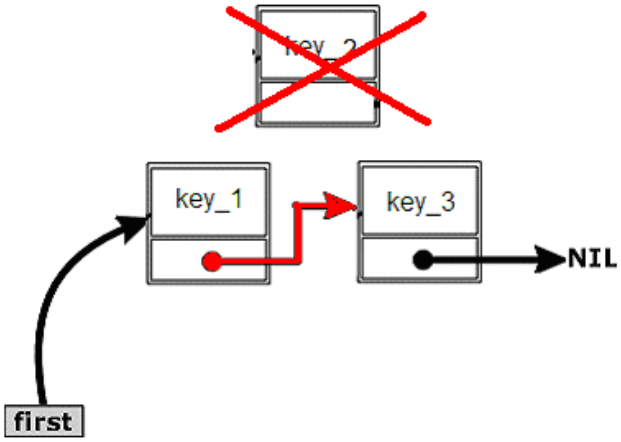
USUWANIE ISTNIEJĄCYCH ELEMENTÓW:

a) z początku

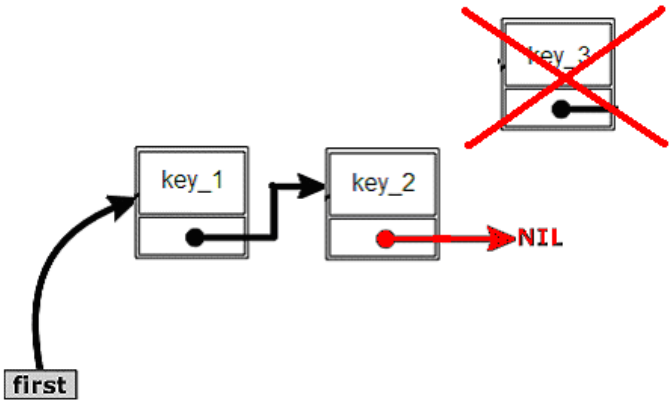
(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.



b) ze środka (usuwamy 2 element)



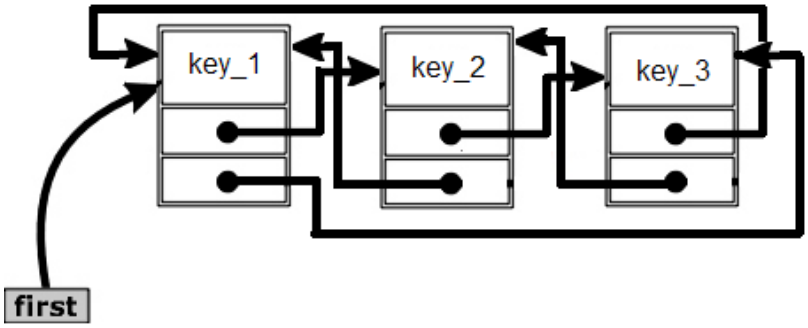
c) z końca



- *Dwukierunkowa*

Lista dwukierunkowa różni się od listy jednokierunkowej tylko tym, że każdy element w swojej strukturze oprócz klucza identyfikacyjnego posiada dwa wskaźniki *next* (wskazuje na następny element na liście) i *previous* (wskazuje na poprzedni element na liście) – dla porównania lista jednokierunkowa posiada tylko wskaźnik *next*.

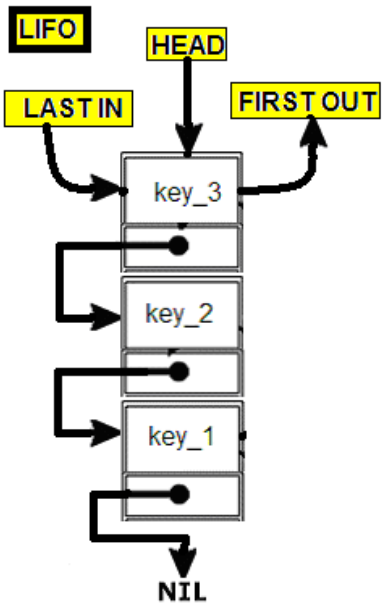
(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.



Na liście dwukierunkowej można wykonywać operacje dodawania, usuwania elementów oraz przeszukiwania tej struktury w sposób analogiczny jak w przypadku listy jednokierunkowej. Trzeba tylko zwrócić baczną uwagę na fakt **zapewnienia spójności podczas dodawania lub usuwania elementów dla dwóch wskaźników**, a nie jak w przypadku listy jednokierunkowej dla jednego. Posiadanie dwóch wskaźników przez każdy element na liście powoduje, że brak ograniczenia związanego ze sposobem przeszukiwania listy (tylko od korzenia do liścia w przypadku listy jednokierunkowej), więc można się przesuwać po liście w obu kierunkach i z dowolnego miejsca, którego wskaźnik jest znany. Powyższy fakt powoduje, że struktura ta jest znacznie efektywniejsza pod względem przeszukiwania, ale niestety więcej czasu potrzebne jest na zarządzanie nowymi elementami. W formie ćwiczeń należy spróbować przerysować powyższe diagramy w taki sposób, aby odpowiadały wykonywanym operacjom dla listy dwukierunkowej.

STOS - LIFO (ANG. LAST IN FIRST OUT)

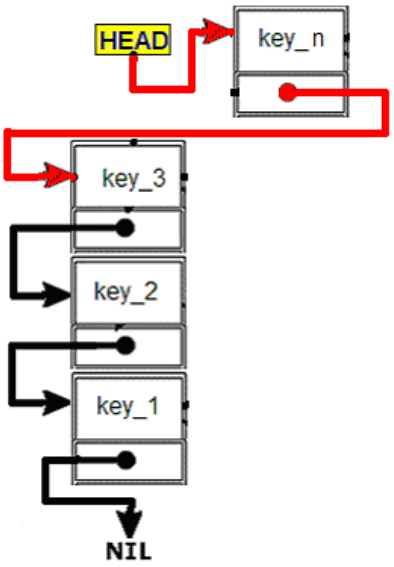
Stos jest szczególnym przypadkiem listy, w którym jedynie ostatni element, zwany *wierzchołkiem (ang. head)*, jest w danym momencie dostępny. Każde **dodanie** nowego elementu powoduje, że staje on się wierzchołkiem, który posiada wskaźnik na element, który był poprzednim wierzchołkiem. Jedynym elementem, który może być **usunięty** bez problemu w tej strukturze to wierzchołek. Jeżeli chce się usunąć element ze środka stosu to należy zdjąć wszystkie elementy (np.: na inny stos), które spowodują, że ten element, który chcemy usunąć stanie się wierzchołkiem, usunąć go a następnie przełożyć ponownie wszystkie elementy z tego tymczasowego stosu na ten właściwy.



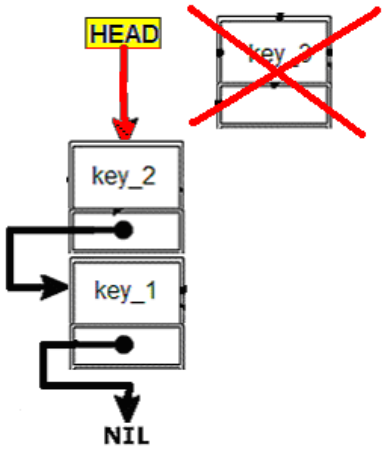
Podczas usuwania rekordu będącego wierzchołkiem należy zapewnić, że nowym wierzchołkiem zostanie rekord, na który usuwany wierzchołek wskazywał jako na następny. Stos jest bardzo często wykorzystywaną strukturą danych. Działanie na nim jest często porównywane do stosu kartek: nie można usunąć kartki znajdującej się na dnie stosu nie usuwając wcześniej wszystkich innych, gdyż cały stos nam się *„rozleci”*. Nie można także dodać nowej kartki gdzieś indziej, niż na samą górę. Stos można wykorzystać w algorytmie zmieniającym notację zapisu liczb z infiksowej na Odwrotną Notację Polską.

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

DODAWANIE NOWEGO ELEMENTU

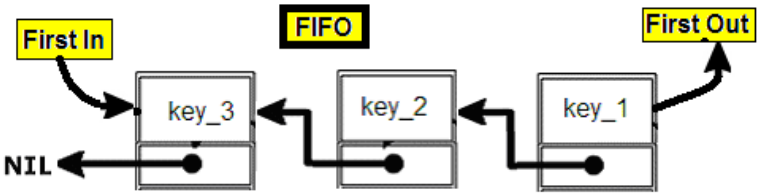


USUWANIE WIERZCHOŁKA STOSU



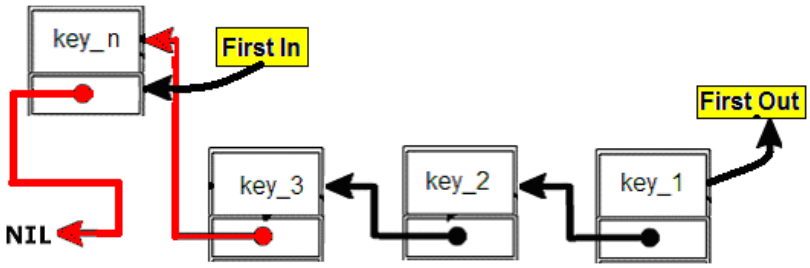
KOLEJKA - FIFO (ANG. FIRST IN FIRST OUT)

Kolejka jest szczególnym przypadkiem listy, w której nowe elementy można jedynie **dodawać na koniec kolejki**, a **usuwać można jedynie z jej początku**.



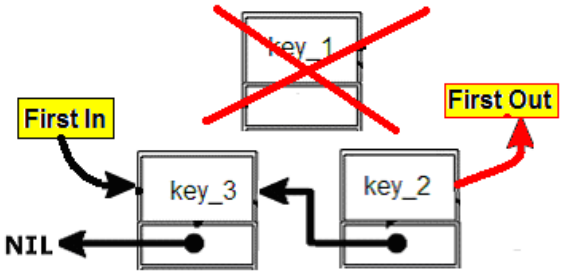
Procedura usunięcia danych z końca kolejki jest podobna, jak w przypadku **stosu**, z tą różnicą, że usuwane są dane od początku a nie od końca. Pierwszy element (a dokładniej wskaźnik do jego miejsca w pamięci) musi zostać zapamiętany, by możliwe było szybkie usuwanie pierwszego elementu. W przeciwnym razie, aby dotrzeć do pierwszego elementu należy przejść całą kolejkę od elementu aktualnego (czyli ostatniego). Możliwe działanie na kolejce są intuicyjnie jasne, gdyż można w prosty sposób skojarzyć ją z kolejką ludzi np.: w sklepie. Każdy nowy klient staje na jej końcu, obsługa odbywa się jedynie na początku.

DODAWANIE NOWEGO ELEMENTU



(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

USUWANIE PIERWSZEGO ELEMENTU KOLEJKI



DRZEWO BST (**B**INARY **S**EARCH **T**REE)

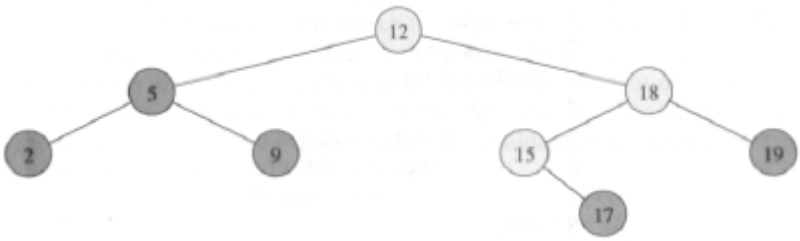
Jest drzewem binarnym, czyli takim, w którym każdy wierzchołek ma co najwyżej dwóch synów. Dodatkowo jest ono drzewem uporządkowanym, tzn. ważna jest kolejność jego synów (w przypadku drzew binarnych pierwszego syna nazywa się *lewym*, a drugiego *prawym*). Charakteryzuje się ono następującą własnością - lewe poddrzewo dowolnego wierzchołka zawiera wierzchołki o mniejszej wartości, a prawe zawiera wierzchołki o większej wartości od jego samego (w przypadku wartości równych wartości wierzchołka można wybrać, do którego drzewa mają należeć, ale trzeba następnie konsekwentnie postępować zgodnie z dokonanym wyborem).

TWORZENIE DRZEWA POPRZEZ DODAWANIE NOWYCH ELEMENTÓW

Stworzenie drzewa *BST* z podanej tablicy należy wykonywać poprzez wstawianie jej kolejnych elementów na to drzewo w taki sposób, aby za każdym spełnione były opisane powyżej zależności Realizuje się to przez wykonywanie następujących czynności dla wszystkich elementów tablicy:

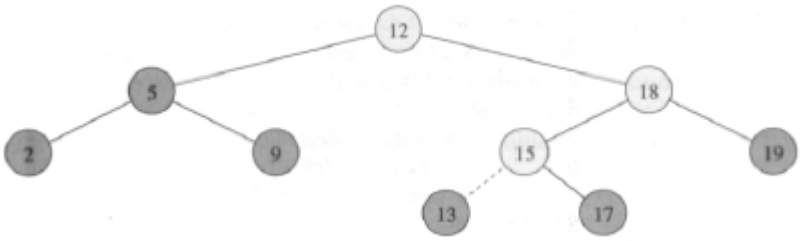
1. Jeśli drzewo *BST* jest puste to wstawiany jest element, który będzie pełnił rolę korzenia (i pomijane są następne punkty i pobierany jest kolejny element z tablicy). Gdy drzewo nie jest puste to porównujemy wstawiany element z korzeniem.
2. Jeżeli wartość elementu jest mniejsza od wartości porównywanego wierzchołka to następuje przejście w głąb drzewa do lewego syna, jeżeli zaś większa to do prawego syna (gdy równa to zależnie od tego jak jaka decyzja została podjęta, byle konsekwentnie).
3. Gdy tam gdzie *"doszliśmy"* nie ma żadnego wierzchołka to przechodzimy do następnego punktu. Jeżeli jednak znajduje się tam inny wierzchołek drzewa to musimy porównać go z wstawianym elementem i wrócić do punktu 2.
4. Skoro doszliśmy do *"wolnego miejsca"* to właśnie tam wstawiamy nasz element. Oczywiście będzie on synem wierzchołka, z którego przyszliśmy (a czy lewym czy prawym to zależy od tego, w którą stronę poszliśmy z tamtego wierzchołka).

Przykład: Załóżmy, że mamy częściowo stworzone drzewo w postaci zaprezentowanej na poniższym rysunku (np.: 12, 18, 5, 19, 2, 15, 9, 17):



(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

Naszym kolejnym zadaniem jest dodanie wartości 13 do powyższego drzewa. W rezultacie uzyskamy drzewo zaprezentowane na poniższym rysunku.



WYSZUKIWANIE ELEMENTÓW

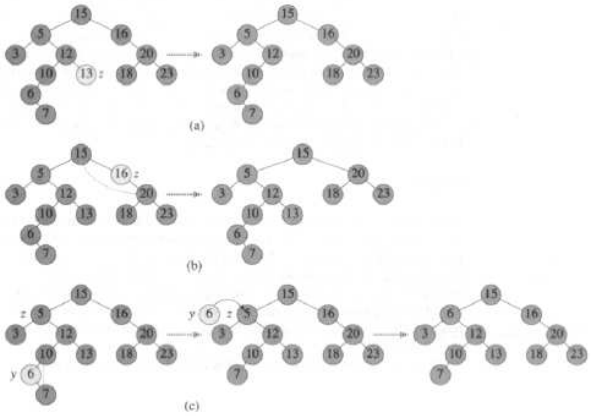
Po zbudowaniu drzewa wyszukanie wierzchołka o konkretnej wartości jest proste. Zaczynamy szukanie od korzenia. Porównujemy szukaną wartość z wartością wierzchołka, na którym się znajdujemy - jeśli jest mniejsza to idziemy głębiej w lewo, jeśli większa to głębiej w prawo. Czynność powtarzamy do czasu, gdy znajdziemy właściwą wartość lub, gdy dojdziemy do wierzchołka, którego nie możemy iść nigdzie głębiej (wtedy nie istnieje w drzewie szukana wartość).

USUWANIE ELEMENTÓW

Poza operacjami wstawiania i wyszukiwania elementu w drzewie *BST* istnieje także operacja usuwania z drzewa (potrzebna dużo rzadziej). Najpierw oczywiście trzeba odnaleźć element do usunięcia, a potem go usunąć. Nie muszę chyba dodawać, że należy to zrobić w taki sposób, aby ciągle były spełnione wszystkie powyżej opisane własności drzewa *BST*. Można wyróżnić trzy przypadki:

1. *Wierzchołek usuwany nie ma synów.*
   Wtedy po prostu go usuwamy.
2. *Wierzchołek usuwany ma jednego syna.*
   Po usunięciu wierzchołka należy jego jedynego syna (z całym jego poddrzewem) *"podczepić"* w miejsce usuwanego wierzchołka. Oznacza to po prostu, że syn usuwanego wierzchołka staje się synem ojca usuwanego wierzchołka.
3. *Wierzchołek usuwany ma dwóch synów.*
   W miejsce usuwanego wierzchołka należy *"podrzucić"* wierzchołek o najmniejszej wartości z prawego poddrzewa usuwanego (lub o największej wartości z lewego poddrzewa). Ten podrzucany wierzchołek jest najbardziej *"lewym"* (lub najbardziej *"prawym"*) w poddrzewie, co daje nam gwarancję, że ma co najwyżej jednego syna i można go usunąć zgodnie z punktem 1 lub 2.

Przykład: Usuwamy wierzchołek *z* drzewa *BST* w trzech możliwych przypadkach: (a) jeżeli wierzchołek *z* nie ma dzieci to go po prostu usuwamy, (b) jeżeli wierzchołek *z* ma tylko jedno dziecko to go usuwamy i podczepiamy jego dziecko w jego miejsce, (c) jeżeli wierzchołek *z* ma dwoje dzieci wtedy podłączamy w jego miejsce jeden z jego następników, który posiada co najwyżej jedno dziecko.



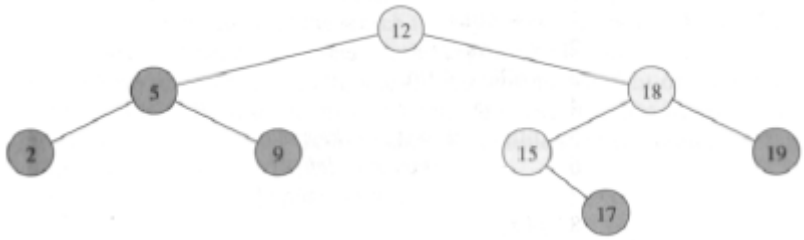
(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

PRZESZUKIWANIE DRZEWA

Przeszukiwanie drzewa jest niczym innym jak podróżą po jego wierzchołkach w odpowiedniej kolejności. Czasem nazywa się to także numerowaniem drzewa, jako że najczęściej przydziela się wierzchołkom numery zgodnie z kolejnością przechodzenia.
Są trzy metody przeszukiwania drzewa:

1. **Pre-order** (*wzdłużna*): korzeń, lewe poddrzewo, prawe poddrzewo.
2. **In-order** (*poprzeczna*): lewe poddrzewo, korzeń, prawe poddrzewo.
3. **Post-order** (*wsteczne*): lewe poddrzewo, prawe poddrzewo, korzeń.

Dla wyjaśnienia skoncentrujmy się na pierwszej metodzie. Najpierw przydzielamy numer korzeniowi naszego drzewa, później przydzielamy kolejne numery całemu lewemu poddrzewu (tzn. wszystkim jego wierzchołkom) i dopiero później całemu prawemu poddrzewu dalsze numery. Przydzielanie numerów poddrzewu wykonuje się tą samą metodą: najpierw korzeniowi (dodam, że korzeń poddrzewa jest synem korzenia naszego drzewa), później jego lewemu poddrzewu i wtedy prawemu. Najlepiej się to wykonuje rekurencyjnie.
Jeśli chodzi o zastosowanie to najprostsze ma metoda druga. Otóż wypisując wierzchołki w kolejności przechodzenia metodą **In-Order** otrzymamy ciąg posortowany. Pozostałe metody oczywiście mają też zastosowanie, ale nieco bardziej zaawansowane (zazwyczaj przydają się do różnych ciekawych algorytmów na grafach).
Przykład: Załóżmy, że mamy stworzone drzewo w postaci zaprezentowanej na poniższym rysunku



Pre-order: 12, 5, 2, 9, 18, 15, 17, 19
In-order: 2, 5, 9, 12, 15, 17, 18, 19
Post-order: 2, 9, 5, 17, 15, 19, 18, 12

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.