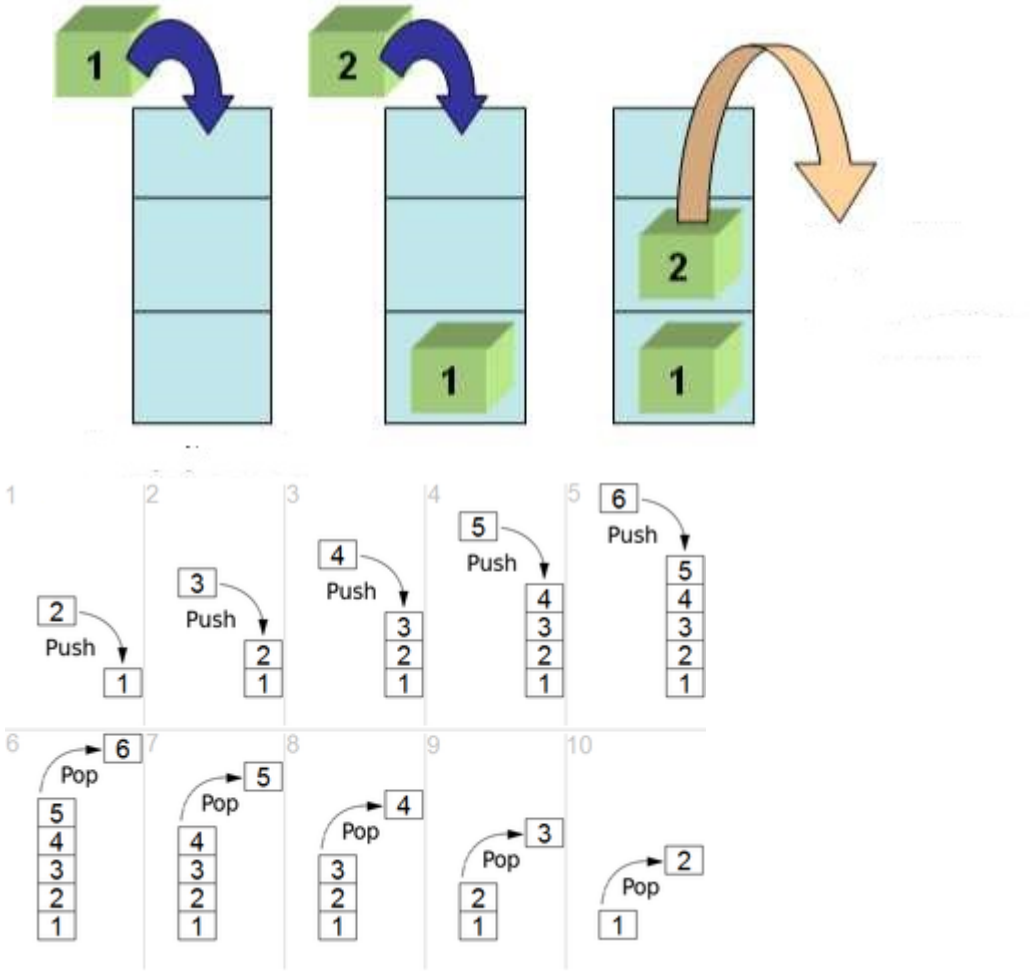
1 Stos: **Stack** i **Stack<T>**

Przykłady z życia:

- Stos talerzy (aby wyciągnąć coś ze środka, musimy wyciągnąć te z góry)
- Meble ładowane do naczepy ciężarówki
- Osoby wsiadające do samolotu i wysiadające z niego.
- Piramida czirliderek (osoba na samej górze schodzi pierwsza)

Zasada: LIFO (ang. Last-In, First-Out), co oznacza: ostatni przyszedł, pierwszy wyszedł.

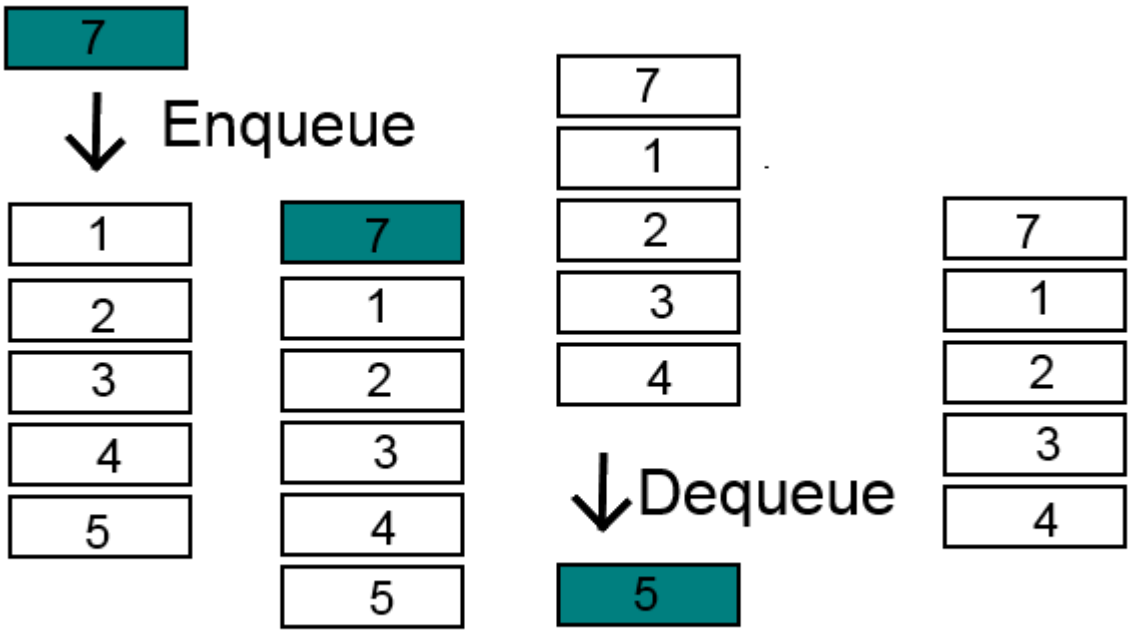


2 Kolejka: `Queue` i `Queue<T>`

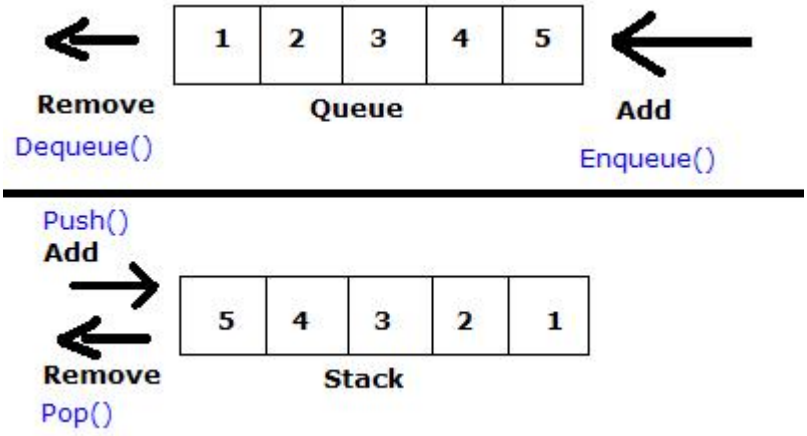
Przykłady z życia:

- samochody poruszające się po jednokierunkowej ulicy,
- ludzie czekający w kolejce,
- klienci czekający na wsparcie techniczne.

Zasada: FIFO (ang. First-In, First-Out), co oznacza: pierwszy przyszedł, pierwszy wyszedł.



2.1 Róznice między stosem a kolejką



3 Lista tablic `ArrayList`

Niegeneryczna kolekcja na przechowywanie tablic obiektów o dynamicznym rozmiarze.

4 Lista `List<T>`

Generyczna kolekcja na przechowywanie tablic obiektów o dynamicznym rozmiarze.

5 Słownik `Dictionary<TKey,TValue>`

Słownik – zbiór wyrazów ułożonych i opracowanych według pewnej zasady, zwykle objaśnianych pod względem znaczeniowym.

6 Słownik posortowany `SortedDictionary<TKey,TValue>`

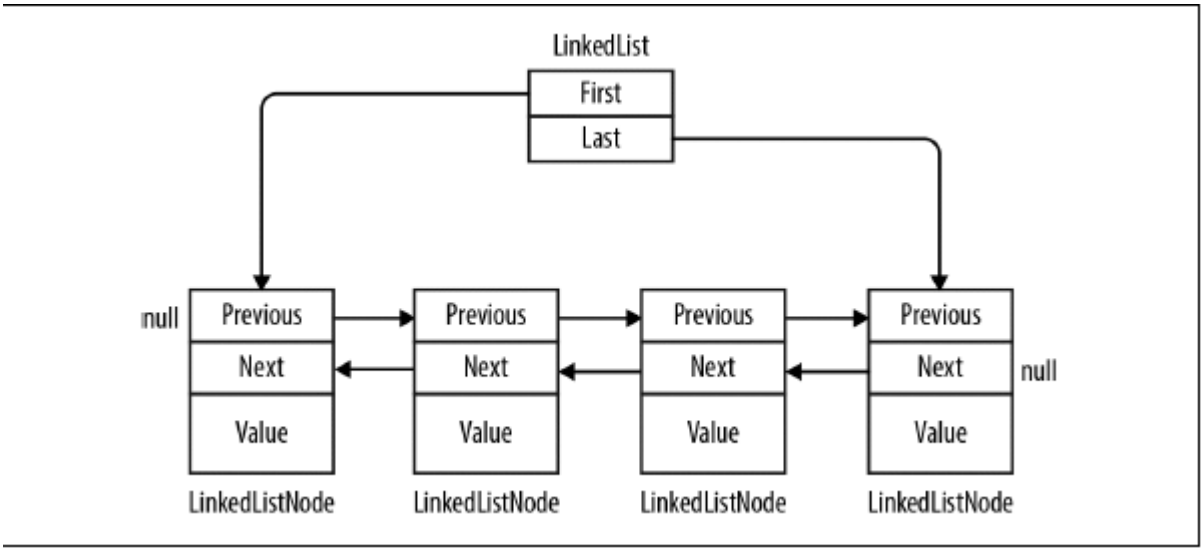
Zachowuje się podobnie jak słownik, z tą różnicą, że kolejne elementy dodawane do słownika są sortowane wg klucza.

7 Lista posortowana `SortedList<TKey,TValue>`

Zachowuje się podobnie jak posortowany słownik.
różnice:

- `SortedList` używa mniej pamięci
- `SortedDictionary` jest szybsze przy operacji wstawiania i usuwania danych
- jeśli elementy są już posortowane, to `SortedList` jest szybsze

8 Lista powiązana/połączona `LinkedList<T>`



9 `BitArray`

Klasa `BitArray` to kolekcja wartości typu `bool` z możliwością dynamicznej zmiany rozmiaru. Pozwala efektywniej wykorzystać pamięć niż zwykła tablica lub struktura `List` wartości typu `bool`, ponieważ do przechowywania każdego elementu potrzebuje tylko jednego bitu, podczas gdy normalnie wartość typu bool zajmuje jeden bajt.

10 `HashSet<T>` i `SortedSet<T>`

Generyczne kolekcje, które wprowadzono odpowiednio w .NET Framework 3.5 i 4.0.

Wspólne cechy:

- Metody `Contains` charakteryzują się dużą szybkością działania dzięki posługiwaniu się algorytmem wyszukiwania wykorzystującym wartości skrótu.
- Nie przechowują duplikatów i niepostrzeżenie ignorują żądania dodania elementów takich samych jak elementy istniejące.
- Nie ma możliwości odwołania się do elementu po jego pozycji.

Klasa `SortedSet<T>` przechowuje elementy w określonym porządku, a `HashSet<T>` nie przechowuje.

11 `Hashtable`

Niegeneryczna wersja klasy `Dictionary<TKey,TValue>`.

Uwaga: lepiej nie tłumaczyć tego na polski jako tablica skrótów. W różnych kontekstach może znaczyć coś innego.

12 `OrderedDictionary`

`OrderedDictionary` to niegeneryczny słownik przechowujący elementy w kolejności ich dodawania. W tej strukturze elementy dostępne są zarówno wg indeksu, jak i wg klucza. `OrderedDictionary` nie jest słownikiem posortowanym. Klasa `OrderedDictionary` jest kombinacją klas `Hashtable` i `ArrayList`, tzn. zawiera całą funkcjonalność pierwszej i kilka dodatkowych funkcji, takich jak `RemoveAt` i indeksator całkowitoliczbowy. Ponadto struktura ta udostępnia własności `Keys` i `Values` zwracające elementy w pierwotnym porządku.

13 ListDictionary

Klasa `ListDictionary` przechowuje dane w liście powiązanej jednostronnie. Nie sortuje elementów, ale zapisuje je w kolejności dodawania. Struktura ta działa bardzo wolno, gdy jest duża. Jedyny sens jej istnienia to wysoka wydajność dla bardzo małych list (zawierających mniej niż dziesięć elementów).

14 HybridDictionary

Klasa `HybridDictionary` to `ListDictionary` automatycznie konwertująca się na `Hashtable` po osiągnięciu określonego rozmiaru w celu uniknięcia problemów wydajnościowych. Chodzi o to, by jak najoszczędniej operować pamięcią, gdy słownik jest mały, oraz by zachować dobrą wydajność, kiedy się powiększy.

15 Collection<T>

Klasa `Collection<T>` to modyfikowalne opakowanie klasy `List<T>`. Mamy dodatkowe metody wirtualne:

```
protected virtual void ClearItems();
protected virtual void InsertItem (int index, T item);
protected virtual void RemoveItem (int index);
protected virtual void SetItem (int index, T item);
```

16 CollectionBase

Niegeneryczna wersja `Collection<T>`.