Bufory

Tablice

- Spójny obszar pamięci podzielony na komórki.
- Komórki w tablicy są numerowane liczbami całkowitymi począwszy od 0.
- Czas dostępu do każdej komórki w tablicy jest stały O(1).
- Tablica jest strukturą nierozszerzalną.
- Zastosowanie: można z góry określić ilość danych; jest mało operacji wstawiania albo usuwania danych ze środka tablicy.

Tablice dynamiczne

- Operacje na tablicy dynamicznej:
  - Dostęp do slotów w obrębie slotów zapełnionych.
  - Dodanie nowej komórki na końcu tablicy (może się to wiązać z utworzeniem nowej dwukrotnie większej tablicy i z przepisaniem danych).
  - Usunięcie komórki z końca tablicy (może się to wiązać z utworzeniem nowej dwukrotnie mniejszej tablicy i z przepisaniem danych).

Tablice dynamiczne

- Idea struktury:
  - Trzymamy tablicę z dodatkowymi parametrami: pojemność (liczba wszystkich slotów w tablicy) i zapełnienie (liczba wykorzystanych slotów w tablicy).
  - Gdy tablica się zapełni w 100%, powiększamy ją dwukrotnie.
  - Gdy zapełnienie tablicy spadnie poniżej 25%, pomniejszamy ją dwukrotnie.

Tablice dynamiczne

- Czasy operacji na tablicy dynamicznej:
  - Dostęp do zapełnionych slotów O(1)
  - Dodanie nowej komórki na końcu tablicy O(n).
  - Usunięcie komórki z końca tablicy O(n).
  - Uwaga: przeważnie dodanie/usunięcie komórki wymaga czasu O(1).
- Analiza zamortyzowana:
  - Za pomocą metody księgowania jednostek kredytowych – każda operacja dodawania/usuwania elementu do/z tablicy pozostawia 2 kredyty czasowe do wykorzystania przy przepisywaniu tablicy.
  - Zamortyzowany koszt każdej pojedynczej operacji na tablicy dynamicznej jest stały O(1) i wynosi 3 jednostki czasowe.

Tablice dynamiczne

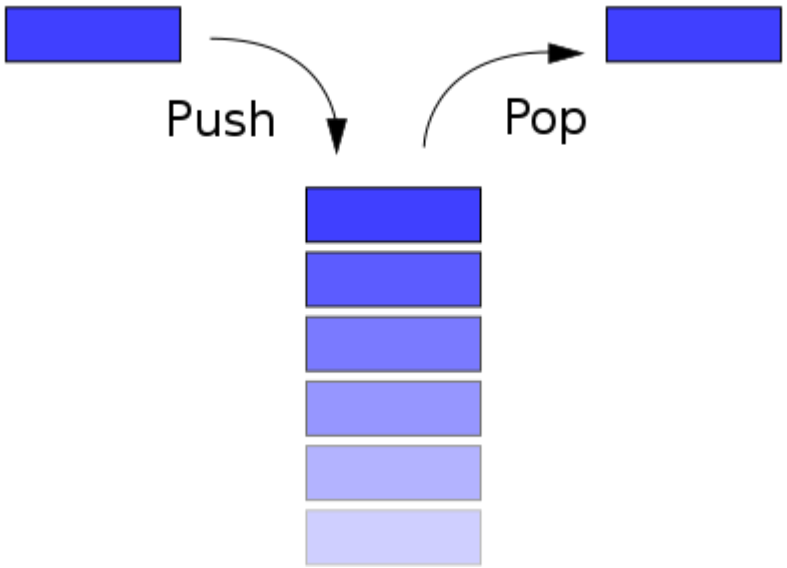
- Zastosowanie: nie można z góry określić rozmiaru danych ale potrzebujemy szybkiego dostępu do każdej komórki w tablicy.
- Realizacja tej struktury w bibliotece standardowej C++: vector<>, deque<>

Stos

- Stos (ang. stack) to struktura, która pozwala na gromadzenie i przetwarzanie danych zgodnie z kolejnością ich wejścia do struktury: element który został najpóźniej dodany do struktury zostanie z niej najszybciej usunięty (ang. LIFO – last in first out , am. LCFS – last come, first served).
- Analogia: stos książek na biurku, ubrania na krześle .

Stos

- Operacje na stosie:
  - Włożenie elementu na szczyt stosu (ang. **push**)
  - Usunięcie elementu ze szczytu stosu (ang. **pop**)
  - Podglądnięcie elementu na szczycie stosu (ang. top)
  - Liczba wszystkich elementów na stosie (ang. size)



Implementacja stosu na tablicy

- Do zwykłej tablicy dokładamy dwie informacje: pojemność stosu (liczba slotów w tablicy), zapełnienie stosu (liczba użytych elementów w tablicy):
  - tab[] – tablica na dane
  - poj – wielkość tablicy
  - zap – zapełnienie tablicy

Implementacja stosu na tablicy

- Operacja push(x):

```
push(x) {
        if (zap = poj) error;
        tab[zap] := x;
        inc(zap);
}
```

- Operacja pop():

```
pop() {
        if (zap = 0) error;
        dec(zap);
        return tab[zap];
}
```

Implementacja stosu na tablicy

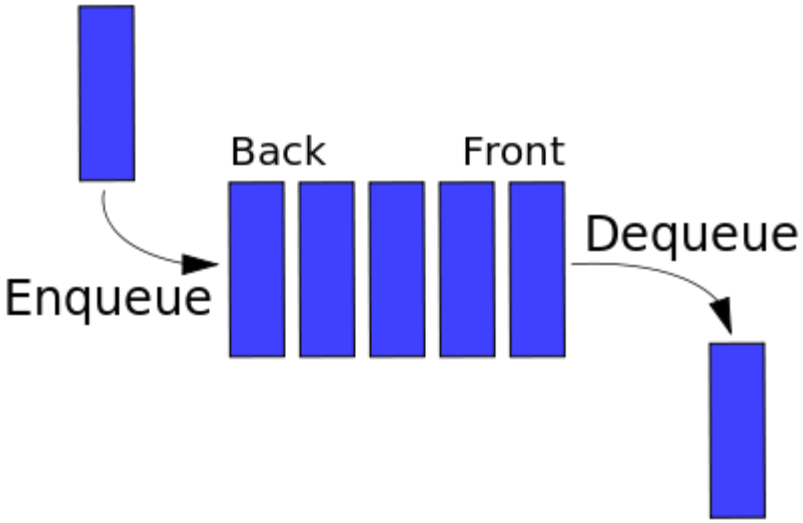
- Wszystkie operacje na stosie zaimplementowanym na tablicy zajmują czas stały O(1).

Kolejka

- Kolejka (ang. queue) to struktura, która pozwala na gromadzenie i przetwarzanie danych zgodnie z kolejnością ich wejścia do struktury: element który został najwcześniej dodany do struktury zostanie z niej najszybciej usunięty (ang. FIFO – first in first out, am. FCFS – first come, first served).
- Analogia: kolejka do kasy.

Kolejka

- Operacje na kolejce:
  - Włożenie elementu na koniec kolejki (ang. **enqueue**)
  - Usunięcie elementu z początku kolejki (ang. **dequeue**)
  - Podglądnięcie elementu na początku kolejki (ang. front)
  - Liczba wszystkich elementów w kolejce (ang. size)



Implementacja kolejki na tablicy

- Do zwykłej tablicy dokładamy trzy informacje: pojemność kolejki (liczba slotów w tablicy), początek kolejki (numer indeksu komórki w którym znajduje się pierwszy element w kolejce), zapełnienie kolejki (liczba użytych elementów w tablicy):
  - tab[] – tablica na dane
  - poj – wielkość tablicy
  - zap – zapełnienie tablicy
  - pocz – miejsce, od którego rozpoczyna się kolejka
- Uwaga: zawijamy tablicę.

Implementacja kolejki na tablicy

- Operacja enqueue(x):

```
enqueue(x) {
        if (zap = poj) error;
        tab[(pocz + zap) mod poj] := x;
        inc(zap);
}
```

- Operacja dequeue():

```
dequeue() {
        if (zap = 0) error;
        pocz = (pocz + 1) mod poj;
        dec(zap);
        return tab[(pocz - 1) mod poj];
}
```

Implementacja kolejki na tablicy

- Wszystkie operacje na kolejce zaimplementowanej na tablicy zajmują czas stały O(1).