Jawaban UAS Struktur Data

Matakuliah: Struktur Data
Dosen: Uro Abdulrohim, MT.

Nama: Rifqi Fadil Fahrial
NIM: 1222646

SOAL 1: CARA KERJA QUEUE MENGGUNAKAN ARRAY

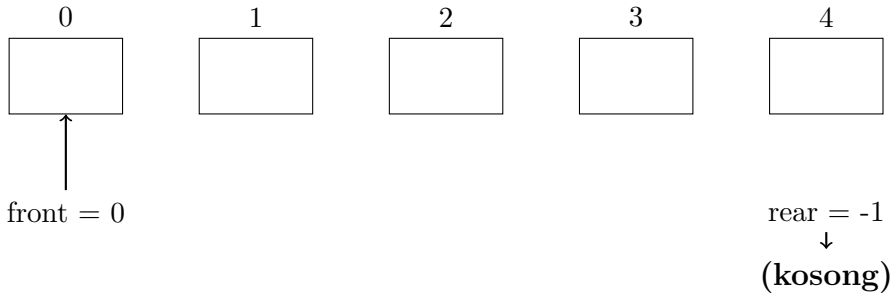
Queue adalah struktur data yang bekerja berdasarkan prinsip FIFO (First In First Out), artinya elemen yang pertama dimasukkan akan keluar lebih dulu.

Untuk implementasi queue menggunakan array, dibutuhkan dua variabel penunjuk:
- `front`: menunjuk ke elemen paling depan.
- `rear`: menunjuk ke elemen paling belakang.

Operasi utama:
- `enqueue(x)`: menambahkan elemen di posisi `rear`.
- `dequeue()`: menghapus elemen dari posisi `front`.

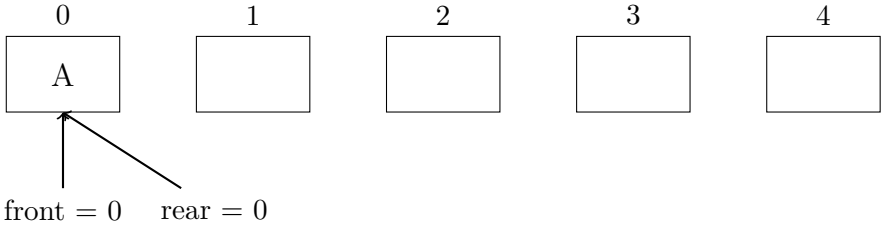
1.1 Ilustrasinya

Pada awalnya, queue kosong, sehingga ‘front = 0‘ dan ‘rear = -1‘.



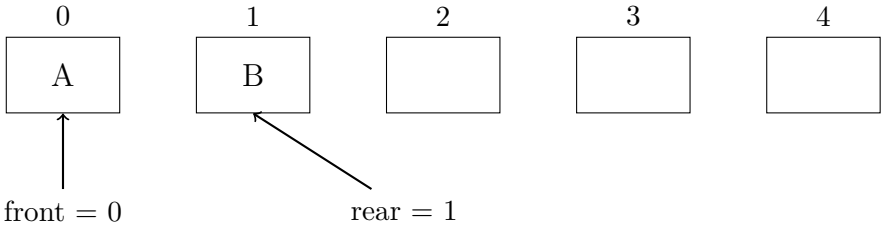
1.2 Setelah Enqueue Pertama (Masukkan ’A’)

Setelah memasukkan elemen pertama (’A’), ‘rear‘ menjadi 0.



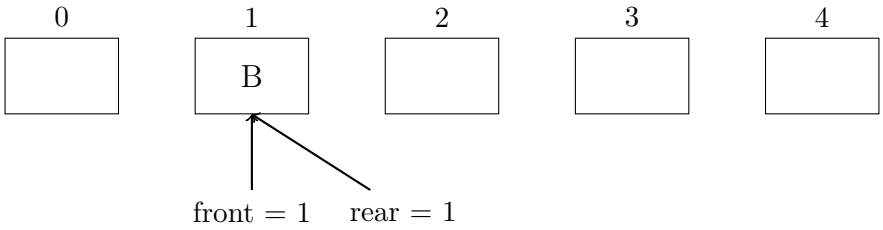
1.3 Setelah Enqueue Kedua (Masukkan 'B')

Setelah memasukkan elemen kedua ('B'), 'rear' menjadi 1.



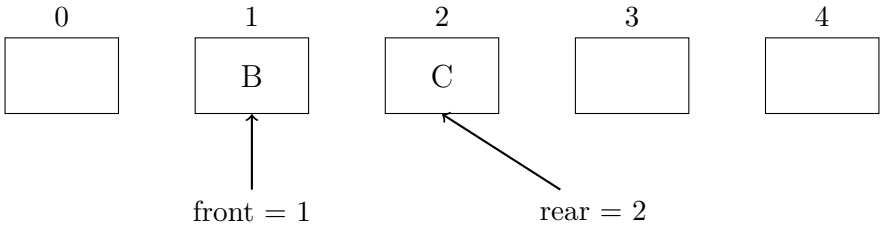
1.4 Setelah Dequeue Pertama (Hapus 'A')

Setelah menghapus elemen pertama ('A'), 'front' menjadi 1.

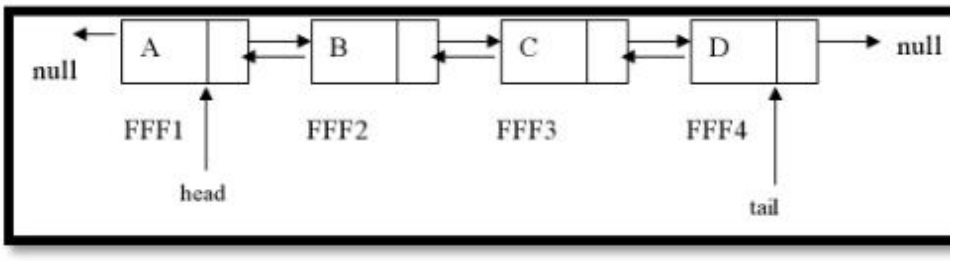


1.5 Setelah Enqueue Ketiga (Masukkan 'C')

Setelah memasukkan elemen ketiga ('C'), 'rear' menjadi 2.



SOAL 2: DEFINISI STRUKTUR DATA DOUBLE LINKED LIST



Struktur data yang ditampilkan di gambar menunjukan sebuah sistem double linked list yang menunjukan sistem pointer yang menghubungkan antar node secara bolak balik sehingga mempercepat pencarian data secara traversal (bolak balik), untuk variabel yang diperlukan adalah

- Data yang disimpan
- Pointer menunjuk ke depan (next)
- pointer menunjuk ke belakang (prev)

Struktur node untuk double linked list:

```
struct Node {
    int data;
    Node* prev;
    Node* next;
};
```

SOAL 3: PROSEDUR INISIALISASI DOUBLE LINKED LIST

untuk menginisialisasi double linked list, maka perlu menambahkan variabel baru yang dapat menunjukkan data yang paling depan (head) dan data paling belakang (tail). maka dapat di inisialisasikan sebagai berikut:

```
void initList(Node*& head, Node*& tail) {
    head = NULL;
    tail = NULL;
}
```

SOAL 4: ALGORITMA PENAMBAHAN DATA DI DEPAN DOUBLE LINKED LIST

Algoritma penambahan di depan:

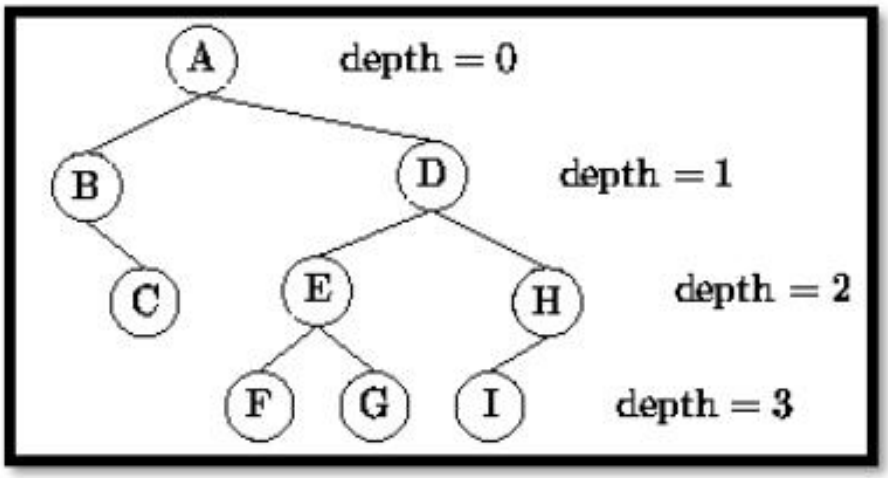
```
void insertDepan(Node*& head, Node*& tail, int nilai) {
    Node* baru = new Node;
    baru->data = nilai;
    baru->prev = NULL;
    baru->next = NULL;

    if (head == NULL) { // Jika list kosong
        head = baru;
        tail = baru;
    } else { // Jika tidak kosong
        baru->next = head;
```

```
        head->prev = baru;
        head = baru;
    }
}
```

SOAL 5: TRAVERSAL PADA TREE (PRE-ORDER, IN-ORDER, POST-ORDER)

Diberikan struktur pohon sebagai berikut:



Hasil traversal:

1. Pre-order Traversal (Root-Left-Right)

- **Aturan:** Kunjungi node akar (Root), lalu kunjungi sub-pohon kiri (Left) secara rekursif, kemudian kunjungi sub-pohon kanan (Right) secara rekursif.
- **Hasilnya** A → B → C → D → E → F → G → H → I

2. In-order Traversal (Left-Root-Right)

- **Aturan:** Kunjungi sub-pohon kiri (Left) secara rekursif, lalu kunjungi node akar (Root), kemudian kunjungi sub-pohon kanan (Right) secara rekursif.
- **Hasilnya** C → B → F → E → G → A → D → H → I

3. Post-order Traversal (Left-Right-Root)

- **Aturan:** Kunjungi sub-pohon kiri (Left) secara rekursif, lalu kunjungi sub-pohon kanan (Right) secara rekursif, kemudian kunjungi node akar (Root).

- **Hasilnya** C → F → G → E → I → H → D → B → A

SOAL 6: DEKLARASI STRUKTUR DATA UNTUK TREE

Struktur node untuk tree:

```
struct Node {
    int data;
    Node* kiri;
    Node* kanan;
};

Node* root = NULL;
```

Variabel yang dibutuhkan adalah :

1. Variabel data yang disimpan, dalam kasus ini adalah integer
2. Pointer yang menunjuk ke node sama yang dalam kasus tree ini mengarah ke arah kiri dan kanan karena ada dua arah yang dapat diakses untuk, dengan value NULL sebagai default yang berarti value kosong / tidak ada / tidak dibuat
3. Variabel root yang mana menginisialisasi kan Tree data structure yang menjadi awal dari pencarian data.

SOAL 7: IMPLEMENTASI CLASS TKALKULATOR DALAM BAHASA C++

Implementasi class `TKalkulator`:

```
#include <iostream>
using namespace std;

class TKalkulator {
private:
    int opr1;
    int opr2;
    int hasil;

public:
    void setBilangan1(int op) { opr1 = op; }
    void setBilangan2(int op) { opr2 = op; }

    int getBilangan1() { return opr1; }
    int getBilangan2() { return opr2; }
```

```
    int getHasil() { return hasil; }

    int Penjumlahan() {
        hasil = opr1 + opr2;
        return hasil;
    }

    int Pengurangan() {
        hasil = opr1 - opr2;
        return hasil;
    }

    int Perkalian() {
        hasil = opr1 * opr2;
        return hasil;
    }

    int PembagianBulat() {
        if (opr2 != 0)
            hasil = opr1 / opr2;
        else {
            cout << "Error: Pembagi nol!" << endl;
            hasil = 0;
        }
        return hasil;
    }

    int PembagianSisa() {
        if (opr2 != 0)
            hasil = opr1 % opr2;
        else {
            cout << "Error: Pembagi nol!" << endl;
            hasil = 0;
        }
        return hasil;
    }
};

int main() {
    TKalkulator calc;
    calc.setBilangan1(10);
    calc.setBilangan2(3);

    cout << "Penjumlahan: " << calc.Penjumlahan() << endl;
    cout << "Pengurangan: " << calc.Pengurangan() << endl;
    cout << "Perkalian: " << calc.Perkalian() << endl;
    cout << "Pembagian Bulat: " << calc.PembagianBulat() << endl;
    cout << "Pembagian Sisa: " << calc.PembagianSisa() << endl;
```

```
    return 0;
}
```