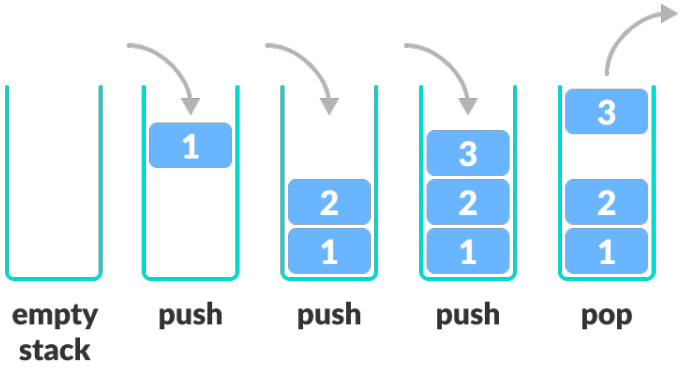
Nama : Isra Nirwana Nur N. Kalau
NIM : H071211036
Program Studi : Sistem Informasi
Tugas Struktur Data

STACK AND QUEUE

A. STACK

1. Pengertian Stack

Stack atau dalam Bahasa Indonesia diartikan tumpukan, adalah struktur data linier yang mengikuti prinsip Last In First Out (LIFO). Artinya elemen yang terakhir disisipkan akan menjadi elemen pertama yang keluar. Dalam istilah pemrograman, upaya menambahkan elemen pada struktur data stack disebut dengan push. Sedangkan proses menghapus atau menghilangkan elemen data dari stack disebut pop.



Sumber: programiz.com

2. Query Stack

Contoh kode implementasi stack dengan bahasa pemrograman C++

```
// Stack implementation in C++

#include <stdlib.h>
#include <iostream>
```

```
using namespace std;

#define MAX 10
int size = 0;

// Creating a stack
struct stack {
  int items[MAX];
  int top;
};
typedef struct stack st;

void createEmptyStack(st *s) {
  s->top = -1;
}

// Check if the stack is full
int isfull(st *s) {
  if (s->top == MAX - 1)
    return 1;
  else
    return 0;
}

// Check if the stack is empty
int isempty(st *s) {
  if (s->top == -1)
    return 1;
  else
    return 0;
}

// Add elements into stack
void push(st *s, int newitem) {
  if (isfull(s)) {
    cout << "STACK FULL";
  } else {
    s->top++;
    s->items[s->top] = newitem;
  }
  size++;
}

// Remove element from stack
void pop(st *s) {
  if (isempty(s)) {
    cout << "\n STACK EMPTY \n";
  } else {
    cout << "Item popped= " << s->items[s->top];
    s->top--;
  }
  size--;
  cout << endl;
}
```

```
// Print elements of stack
void printStack(st *s) {
  printf("Stack: ");
  for (int i = 0; i < size; i++) {
    cout << s->items[i] << " ";
  }
  cout << endl;
}

// Driver code
int main() {
  int ch;
  st *s = (st *)malloc(sizeof(st));

  createEmptyStack(s);

  push(s, 1);
  push(s, 2);
  push(s, 3);
  push(s, 4);

  printStack(s);

  pop(s);

  cout << "\nAfter popping out\n";
  printStack(s);
}
```

Dalam struktur data stack ada dua kondisi yang perlu dihindari, yaitu **underflow** dan **overflow**.

- **Stack underflow**, yaitu keadaan dimana kita mencoba mengakses atau menghapus elemen data pada stack yang kosong
- **Stack overflow**, yaitu keadaan di mana ruang memori yang dialokasikan untuk struktur data stack sudah penuh namun masih dilakukan operasi penyisipan elemen

3. Jenis-Jenis Stack

Berdasarkan kemampuan menyimpan data, struktur data stack dapat dibagi menjadi 2 jenis, yaitu: register stack dan memory stack.

1. Register stack

Register stack merupakan stack yang hanya mampu menampung data dalam jumlah yang kecil. Kedalaman maksimum pada register stack cenderung dibatasi karena ukuran unit memorinya sangat kecil dibandingkan dengan memory stack.

2. Memory stack

Pada stack jenis ini, kedalaman dari stack cukup fleksibel dan mampu menangani dalam dalam skala yang lebih besar dibandingkan jenis sebelumnya.

4. Karakteristik Stack

Struktur data stack memiliki ciri sebagai berikut:

- Stack digunakan pada banyak algoritma yang berbeda seperti Tower of Hanoi, Tree traversal, rekursi dll.
- Stack diimplementasikan dengan struktur data array atau linked list.
- Mengikuti prinsip operasi Last In First Out, yaitu elemen yang dimasukkan pertama akan muncul terakhir dan sebaliknya.
- Penyisipan dan penghapusan terjadi di satu ujung yaitu dari atas tumpukan.
- Apabila ruang memori yang dialokasikan untuk struktur data stack sudah penuh namun masih dilakukan operasi penyisipan elemen maka akan terjadi stack overflow.
- Apabila struktur data tidak memiliki elemen data atau kosong, namun tetap dilakukan operasi penghapusan maka akan terjadi stack underflow.

5. Operasi-Operasi Dasar pada Stack

Ada beberapa operasi dasar yang bisa kita untuk lakukan terhadap struktur data stack. Operasi-operasi tersebut meliputi :

- Push: Menyisipkan elemen ke bagian atas stack
- Pop: Menghapus elemen atas dari stack
- IsEmpty: Memeriksa apakah stack kosong

- IsFull: Memerika apakah stack sudah penuh
- Peek: Mendapatkan nilai elemen teratas tanpa menghapusnya

6. Fungsi dan Kegunaan Stack

Adapun fungsi dan kegunaan struktur data stack adalah sebagai berikut:

- Struktur data stack digunakan dalam evaluasi dan konversi ekspresi aritmatika. Proses ini banyak dipakai untuk program kompiler.
- Stack digunakan dalam pemrograman rekursi.
- Digunakan untuk pemeriksaan tanda kurung.
- Stack digunakan dalam manajemen memori.
- Dipakai untuk memproses pemanggilan sebuah fungsi.

7. Kelebihan dan Kekurangan Menggunakan Stack

Adapun kelebihan menggunakan struktur data stack di antaranya:

- **Manajemen data yang efisien**: Stack membantu mengelola data berdasarkan prinsip operasi LIFO yang tidak bisa dilakukan dengan linked list dan array.
- **Manajemen fungsi yang efisien**: Ketika suatu fungsi dipanggil, variabel lokal disimpan dalam stack, dan secara otomatis dihancurkan setelah dikembalikan.
- **Kontrol atas memori**: Stack memungkinkan kita untuk mengontrol bagaimana memori dialokasikan dan tidak dialokasikan.
- **Manajemen memori cerdas**: Stack secara otomatis membersihkan objek.
- **Tidak mudah rusak:** Stack tidak mudah rusak, oleh karena itu stack cenderung lebih aman dan dapat diandalkan.

- **Tidak mengizinkan pengubahan ukuran variabel**: Variabel pada stack tidak dapat diubah ukurannya.

Selain kelebihan di atas, stack juga terdapat beberapa kelemahan berikut:

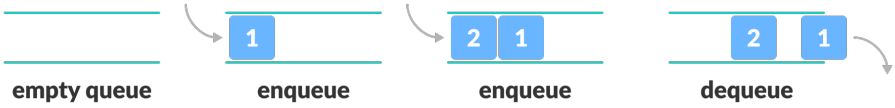
- **Ukuran memori terbatas**: Memori pada stack cukup terbatas.
- **Kemungkinan stack overflow**: Terlalu banyak membuat objek di stack dapat meningkatkan risiko stack overflow.
- **Akses acak tidak dimungkinkan**: Dalam stack, akses data secara acak tidak bisa dilakukan. Data yang dapat diakses adalah data yang berada pada elemen atas.
- **Dapat menyebabkan fungsi tidak tedefinisi:** Ketika penyimpanan variabel akan ditimpa, kadang-kadang akan menyebabkan perilaku fungsi atau program yang tidak terdefinisi.
- **Penghentian yang tidak diinginkan**: Jika stack berada di luar memori maka dapat menyebabkan penghentian yang tidak normal.

B. QUEUE

1. Pengertian Queue

Queue adalah struktur data linier yang menerapkan prinsip operasi dimana elemen data yang masuk pertama akan keluar lebih dulu. Prinsip ini dikenal dengan istilah FIFO (First In, First Out). Berbeda dengan struktur data stack yang menyimpan data secara bertumpuk dimana hanya terdapat satu ujung yang terbuka untuk melakukan operasi data, struktur data queue justru disusun secara horizontal dan terbuka di kedua ujungnya. Ujung pertama (head) digunakan untuk menghapus data sedangkan ujung lainnya (tail) digunakan untuk menyisipkan data. Persamaan antara stack dan queue

adalah keduanya dapat diimplementasikan menggunakan struktur data linked list atau array.



Sumber: naukri.com

2. Query Queue

Berikut adalah implementasi queue dengan bahasa pemrograman C++ :

```
// Queue implementation in C++

#include <iostream>
#define SIZE 5

using namespace std;

class Queue {
   private:
  int items[SIZE], front, rear;

   public:
  Queue() {
    front = -1;
    rear = -1;
  }

  bool isFull() {
    if (front == 0 && rear == SIZE - 1) {
      return true;
    }
    return false;
  }

  bool isEmpty() {
    if (front == -1)
      return true;
    else
      return false;
  }

  void enQueue(int element) {
    if (isFull()) {
      cout << "Queue is full";
    } else {
      if (front == -1) front = 0;
      rear++;
      items[rear] = element;
```

```
      cout << endl
         << "Inserted " << element << endl;
    }
  }

  int deQueue() {
    int element;
    if (isEmpty()) {
      cout << "Queue is empty" << endl;
      return (-1);
    } else {
      element = items[front];
      if (front >= rear) {
        front = -1;
        rear = -1;
      } /* Q has only one element, so we reset the queue
after deleting it. */
      else {
        front++;
      }
      cout << endl
         << "Deleted -> " << element << endl;
      return (element);
    }
  }

  void display() {
    /* Function to display elements of Queue */
    int i;
    if (isEmpty()) {
      cout << endl
         << "Empty Queue" << endl;
    } else {
      cout << endl
         << "Front index-> " << front;
      cout << endl
         << "Items -> ";
      for (i = front; i <= rear; i++)
        cout << items[i] << "  ";
      cout << endl
         << "Rear index-> " << rear << endl;
    }
  }
};

int main() {
  Queue q;

  //deQueue is not possible on empty queue
  q.deQueue();

  //enQueue 5 elements
  q.enQueue(1);
  q.enQueue(2);
  q.enQueue(3);
  q.enQueue(4);
```

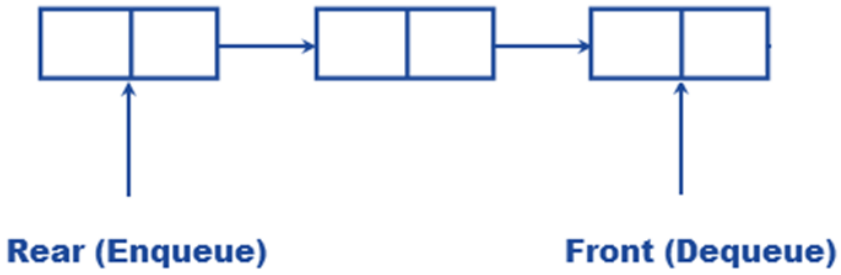
```
    q.enQueue(5);

    // 6th element can't be added to because the queue is full
    q.enQueue(6);

    q.display();

    //deQueue removes element entered first i.e. 1
    q.deQueue();

    //Now we have just 4 elements
    q.display();

    return 0;
}
```

3. Jenis-Jenis Queue

Secara umum ada 4 jenis struktur data queue, meliputi

1. Simple Queue

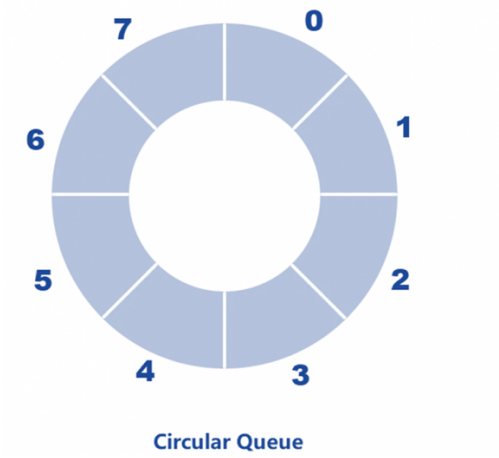
Simple queue adalah struktur data queue paling dasar di mana penyisipan item dilakukan di simpul belakang (**rear** atau **tail**) dan penghapusan terjadi di simpul depan (**front** atau **head**).



Sumber: naukri.com

2. Circular Queue

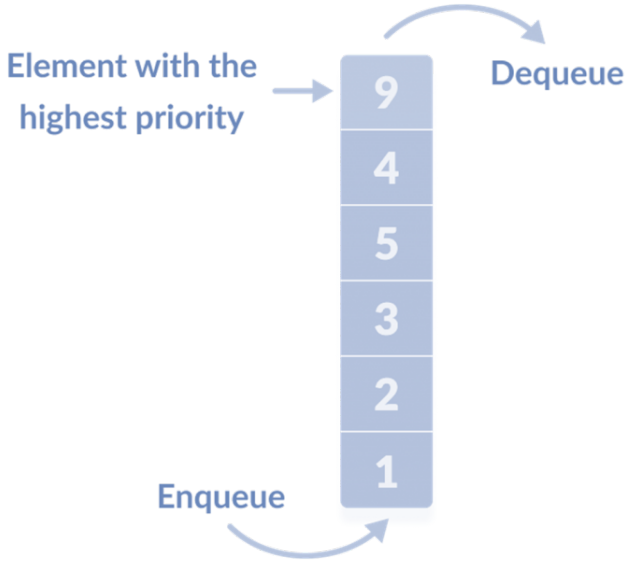
Pada circular queue, simpul terakhir terhubung ke simpul pertama. Queue jenis ini juga dikenal sebagai Ring Buffer karena semua ujungnya terhubung ke ujung yang lain. Penyisipan terjadi di akhir antrian dan penghapusan di depan antrian.



Sumber: naukri.com

3. **Priority Queue**

Priority Queue adalah strruktur data queue dimana simpul akan memiliki beberapa prioritas yang telah ditentukan. Simpul dengan prioritas terbesar akan menjadi yang pertama dihapus dari antrian. Sedangkan penyisipan item terjadi sesuai urutan kedatangannya.



Sumber: naukri.com

4. **Double-Ended Queue (Dequeue)**

Dalam double-ended queue (dequeue), operasi penyisipan dan penghapusan dapat terjadi di ujung depan dan belakang dari queue.

Sumber: naukri.com

4. Karakteristik Queue

Queue memiliki berbagai karakteristik sebagai berikut:

- Queue adalah struktur FIFO (First In First Out).
- Untuk menghapus elemen terakhir dari Queue, semua elemen yang dimasukkan sebelum elemen tersebut harus dihilangkan atau dihapus.
- Queue adalah daftar berurutan dari elemen-elemen dengan tipe data yang serupa.

5. Operasi-Operasi Dasar pada Queue

Queue adalah struktur data abstrak (ADT) yang memungkinkan operasi berikut:

- Enqueue: Menambahkan elemen ke akhir antrian
- Dequeue: Menghapus elemen dari depan antrian
- IsEmpty: Memeriksa apakah antrian kosong
- IsFull: Memeriksa apakah antrian sudah penuh
- Peek: Mendapatkan nilai bagian depan antrian tanpa menghapusnya
- Initialize: Membuat antrian baru tanpa elemen data (kosong)

Namun, secara umum antrian memiliki 2 operasi utama, yaitu enqueue dan dequeue.

1. Operasi Enqueue

Di bawah ini adalah langkah-langkah untuk enqueue (memasukkan) data ke dalam antrian :

- Periksa apakah antrian sudah penuh atau tidak.
- Jika antrian penuh – cetak kesalahan overflow dan keluar dari program.

- Jika antrian tidak penuh – naikkan pointer belakang untuk menunjuk ke ruang kosong berikutnya.
- Tambahkan elemen pada posisi yang ditunjuk oleh pointer belakang.
- Kembalikan status bahwa penambahan telah berhasil

Pseucode untuk operasi enqueue :

```
procedure enqueue (data)

if queue is full
return overflow
endif
rear ← rear + 1
queue[rear] ← data
return true
end procedure
```

2. **Operasi Dequeue**

Di bawah ini adalah langkah-langkah untuk melakukan operasi dequeue :

- Periksa apakah antrian sudah penuh atau tidak.
- Jika antrian kosong – cetak kesalahan underflow dan keluar dari program.
- Jika antrian tidak kosong – akses elemen data yang ditunjuk oleh pointer depan.
- Geser pointer depan untuk menunjuk ke elemen data berikutnya yang tersedia.
- Kembalikan status bahwa operasi penghapusan telah berhasil

Pseucode untuk operasi dequeue :

```
procedure dequeue

if queue is empty
return underflow
```

```
end if
data = queue[front]
front ← front + 1
return true
end procedure
```

6. Fungsi dan Kegunaan Queue

Berikut ini adalah beberapa fungsi queue yang paling umum dalam struktur data:

- Queue banyak digunakan untuk menangani lalu lintas (traffic) situs web.
- Membantu untuk mempertahankan playlist yang ada pada aplikasi media player
- Queue digunakan dalam sistem operasi untuk menangani interupsi.
- Membantu dalam melayani permintaan pada satu sumber daya bersama, seperti printer, penjadwalan tugas CPU, dll.
- Digunakan dalam transfer data asinkronus misal pipeline, IO file, dan socket.

7. Kelebihan dan Kekurangan Queue

Kelebihan queue di antarnya:

- Data dalam jumlah besar dapat dikelola secara efisien.
- Operasi seperti penyisipan dan penghapusan dapat dilakukan dengan mudah karena mengikuti aturan masuk pertama keluar pertama.
- Queue berguna ketika layanan tertentu digunakan oleh banyak konsumen.
- Queue cepat untuk komunikasi antar-proses data.
- Queue dapat digunakan dalam implementasi struktur data lainnya.

Kelemahan struktur data queue adalah sebagai berikut:

- Operasi seperti penyisipan dan penghapusan elemen dari tengah cenderung banyak memakan waktu.
- Dalam queue konvensional, elemen baru hanya dapat dimasukkan ketika elemen yang ada dihapus dari antrian.
- Mencari elemen data pada struktur queue membutuhkan time complexity O(N).
- Ukuran maksimum antrian harus ditentukan sebelumnya