Modul Praktikum
SDAA

4

STACK DAN QUEUE

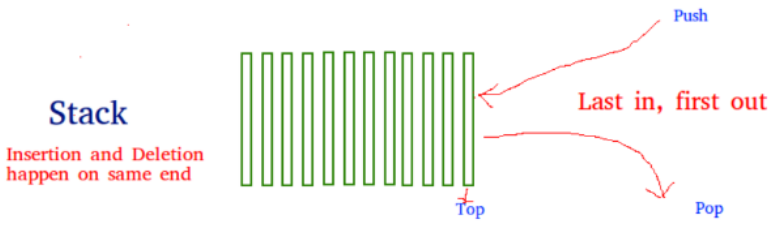
STACK

Stack atau tumpukan adalah struktur data linier yang menggunakan prinsip **LIFO (Last In First Out)**. Secara sederhana, stack dapat diibaratkan sebagai tumpukan buku yang disusun ke atas. Buku pertama yang ditumpuk akan berada di bagian paling bawah, sementara buku-buku berikutnya diletakkan di atasnya. Saat mengambil buku, yang pertama diambil adalah buku yang terakhir ditumpuk. Berikut adalah beberapa operasi yang dapat dilakukan pada stack:

1. **Push**
   Menambah sebuah item pada stack, jika penambahan dilakukan pada kondisi stack penuh maka kondisi ini disebut overflow.

2. **Pop**
   Menghapus sebuah item pada stack , hal ini dilakukan pada tumpukan yang paling atas, jika penghapusan dilakukan pada saat kondisi stack kosong maka kondisi ini disebut underflow.

3. **Peek or Top**
   Melihat item teratas pada stack.



Gambar 1. Stack

SOURCE CODE IMPLEMENTASI STACK MENGGUNAKAN ARRAY

Berikut adalah source code untuk melakukan implementasi Stack pada struktur data Array. Code dapat dilihat pada link berikut: https://pastebin.com/mWpTcjR5.

```
#include <iostream>
#define MAX 100
using namespace std;

int stack[MAX];
int top = -1; // Menandakan elemen teratas pada stack

void push(int value) {
```

```
    if (top >= MAX - 1) {
        cout << "Stack overflow" << endl;
    } else {
        top++;
        stack[top] = value;
        cout << value << " pushed into stack" << endl;
    }
}

int pop() {
    if (top < 0) {
        cout << "Stack underflow" << endl;
        return -1;
    } else {
        int value = stack[top];
        top--;
        return value;
    }
}

// Melihat elemen teratas pada stack
int peek() {
    if (top < 0) {
        cout << "Stack is empty" << endl;
        return -1;
    } else {
        return stack[top];
    }
}

bool isEmpty() {
    return (top < 0);
}

int main() {
    push(10);
    push(20);
    push(30);

    cout << "Top element is " << peek() << endl;

    cout << pop() << " popped from stack" << endl;
    cout << pop() << " popped from stack" << endl;

    if (isEmpty()) {
        cout << "Stack is empty" << endl;
    } else {
        cout << "Stack is not empty" << endl;
    }

    return 0;
}
```

SOURCE CODE IMPLEMENTASI STACK MENGGUNAKAN LINKED LIST

Berikut adalah source code untuk melakukan implementasi Stack pada struktur data Linked List. Code dapat dilihat pada link berikut: https://pastebin.com/VKnh1r2v.

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

struct Mahasiswa {
    string nama;
    int nim;
    double ipk;
};

struct Node {
    Mahasiswa data;
    Node* next;
};

Node* createNode() {
    Node* newNode = new Node();
    cout << "Masukan nama : ";
    cin >> newNode->data.nama;
    cout << "Masukan nim  : ";
    cin >> newNode->data.nim;
    cout << "Masukan ipk  : ";
    cin >> newNode->data.ipk;
    return newNode;
}

void push(Node** top) {
    Node* nodeBaru = createNode();
    nodeBaru->next = *top;
    *top = nodeBaru;
}

void pop(Node** top) {
    if (*top == NULL) {
        cout << "Stack underflow\n";
        return;
    }
    Node* temp = *top;
    *top = (*top)->next;
    delete temp;
}

void display(Node* top) {
    if (!top) {
        cout << "Stack kosong\n";
        return;
```

```
    }
    while (top) {
        cout << "Nama : " << top->data.nama << "\n";
        cout << "NIM  : " << top->data.nim << "\n";
        cout << "IPK  : " << top->data.ipk << "\n";
        cout << "----------------------------\n";
        top = top->next;
    }
}

int main() {
    Node* TOP = NULL;
    int menu;

    do {
        cout << "1. Push\n2. Pop\n3. Display\n4. Exit\nMasukan pilihan: ";
        cin >> menu;
        switch (menu) {
            case 1: push(&TOP); break;
            case 2: pop(&TOP); break;
            case 3: display(TOP); break;
            case 4: break;
            default: cout << "Pilihan tidak ada\n";
        }
    } while (menu != 4);

    return 0;
}
```

QUEUE

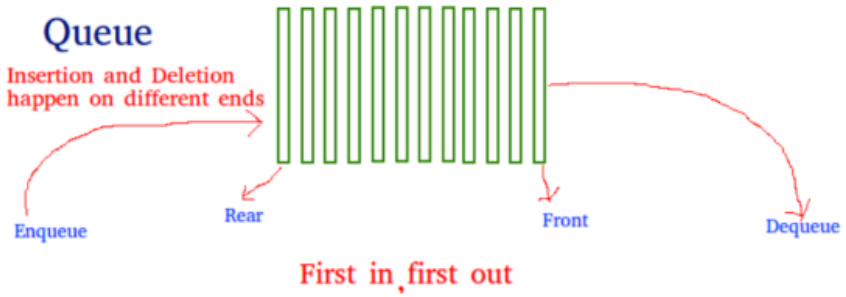
Queue atau antrian dalam struktur data adalah kumpulan elemen di mana penambahan elemen hanya dapat dilakukan pada salah satu ujung yang disebut sisi belakang (rear), sementara penghapusan elemen dilakukan melalui ujung lain yang disebut sisi depan (front). Prinsip dasar dari queue adalah **FIFO (First In First Out)**, yang berarti elemen yang pertama kali dimasukkan akan menjadi yang pertama diakses atau dikeluarkan. Sebagai analogi, hal ini serupa dengan antrian di loket pembelian tiket kereta, di mana orang yang datang lebih dulu akan dilayani lebih dahulu, dan akan selesai lebih cepat daripada orang yang datang belakangan. Berikut adalah beberapa operasi yang dapat dilakukan pada queue:

1. **Enqueue**
   Penambahan item pada queue, penambahan dilakukan pada ekor antrian jika penambahan dilakukan pada saat queue penuh maka kondisi ini disebut overflow.

2. **Dequeue**
   Penghapusan item pada queue, penghapusan dilakukan pada kepala (Front) antrian, jika penghapusan dilakukan pada saat queue kosong maka kondisi ini disebut underflow.

3. **Front**
   Mengambil item antrian pertama pada queue.

4. **Rear**
   Mengambil item antrian terakhir pada queue.



Gambar 2. Queue

SOURCE CODE IMPLEMENTASI QUEUE MENGGUNAKAN ARRAY

Berikut adalah source code untuk melakukan implementasi Queue pada struktur data Array. Code dapat dilihat pada link berikut: https://pastebin.com/dJz1AXTF.

```
#include <iostream>
#define MAX 100

using namespace std;

int queue[MAX];
int front = -1, rear = -1;

bool isFull() {
    return rear == MAX - 1;
}

bool isEmpty() {
    return front == -1 || front > rear;
}

void enqueue(int value) {
    if (isFull()) {
        cout << "Queue overflow" << endl;
        return;
    }
    if (front == -1) {
        front = 0;  // Mengubah variabel depan menjadi 0 apabila elemen
merupakan elemen pertama
```

```
    }
    rear++;
    queue[rear] = value;
    cout << value << " enqueued into queue" << endl;
}

int dequeue() {
    if (isEmpty()) {
        cout << "Queue underflow" << endl;
        return -1;
    }
    int value = queue[front];
    front++;
    if (isEmpty()) {  // Reset apabila queue menjadi kosong
        front = rear = -1;
    }
    return value;
}

int peek() {
    if (isEmpty()) {
        cout << "Queue is empty" << endl;
        return -1;
    }
    return queue[front];
}

void display() {
    if (isEmpty()) {
        cout << "Queue is empty" << endl;
        return;
    }
    cout << "Queue elements: ";
    for (int i = front; i <= rear; i++) {
        cout << queue[i] << " ";
    }
    cout << endl;
}

int main() {
    enqueue(10);
    enqueue(20);
    enqueue(30);

    cout << "Front element is " << peek() << endl;

    cout << dequeue() << " dequeued from queue" << endl;
    cout << dequeue() << " dequeued from queue" << endl;

    display();

    return 0;
}
```

SOURCE CODE IMPLEMENTASI QUEUE MENGGUNAKAN LINKED LIST

Berikut adalah source code untuk melakukan implementasi Queue pada struktur data Array. Code dapat dilihat pada link berikut: https://pastebin.com/RQZSvn85.

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

struct Mahasiswa {
    string nama;
    int nim;
    double ipk;
};

struct Node {
    Mahasiswa data;
    Node* next;
};

Node* createNode();
void display(Node* front);
void enqueue(Node** front, Node** rear);
void dequeue(Node** front);

int main() {
    Node *FRONT = NULL, *REAR = NULL;
    int menu;

    while (menu != 4) {
        cout << ">> Program antrian bansos <<" << endl;
        cout << "1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\nMasukan pilihan:
";
        cin >> menu;
        switch (menu) {
            case 1: enqueue(&FRONT, &REAR); break;
            case 2: dequeue(&FRONT); break;
            case 3: display(FRONT); break;
            case 4: break;
            default: cout << "Pilihan tidak ada\n";
        }
    }
    return 0;
}

Node* createNode() {
    Node* newNode = new Node();
    cout << "Masukan nama: "; cin >> newNode->data.nama;
    cout << "Masukan nim : "; cin >> newNode->data.nim;
    cout << "Masukan ipk : "; cin >> newNode->data.ipk;
    newNode->next = NULL;
```

```
    return newNode;
}

void display(Node* front) {
    if (!front) {
        cout << "Antrian Kosong\n";
        return;
    }
    while (front) {
        cout << "Nama: " << front->data.nama << "\nNIM : " << front->data.nim <<
"\nIPK : " << front->data.ipk << "\n";
        cout << "----------------------------\n";
        front = front->next;
    }
}

void enqueue(Node** front, Node** rear) {
    Node* nodeBaru = createNode();
    if (!*front) {
        *front = nodeBaru;
    } else {
        (*rear)->next = nodeBaru;
    }
    *rear = nodeBaru;
}

void dequeue(Node** front) {
    if (!*front) {
        cout << "Antrian Kosong\n";
        return;
    }
    Node* temp = *front;
    *front = (*front)->next;
    delete temp;
}
```