

LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA
MODUL 7
QUEUE



Disusun oleh
Faqih Abdullah
2311102048

Dosen Pengampu
Wahyu Andi Saputra, SPd., M.Eng

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKUTLAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024

A. DASAR TEORI

Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode **FIFO** (First-In First-Out). Data yang pertama dimasukkan ke dalam queue akan menjadi data yang pertama pula untuk dikeluarkan dari queue. Queue mirip dengan konsep **antrian** pada kehidupan sehari-hari, dimana konsumen yang datang lebih dulu akan dilayani terlebih dahulu.

Implementasi queue dapat dilakukan dengan menggunakan array atau linked list. Struktur data queue terdiri dari dua pointer yaitu front dan rear. **Front/head** adalah pointer ke elemen pertama dalam queue dan **rear/tail/back** adalah pointer ke elemen terakhir dalam queue.



FIRST IN FIRST OUT (FIFO)

Perbedaan antara stack dan queue terdapat pada aturan penambahan dan penghapusan elemen. Pada stack, operasi penambahan dan penghapusan elemen dilakukan di satu ujung. Elemen yang terakhir diinputkan akan berada paling dengan dengan ujung atau dianggap paling atas sehingga pada operasi penghapusan, elemen teratas tersebut akan dihapus paling awal, sifat demikian dikenal dengan LIFO.

Pada Queue, operasi tersebut dilakukan ditempat berbeda (melalui salah satu ujung) karena perubahan data selalu mengacu pada Head, maka hanya ada 1 jenis insert maupun delete. Prosedur ini sering disebut **Enqueue** dan **Dequeue** pada kasus Queue. Untuk Enqueue, cukup tambahkan elemen setelah elemen terakhir Queue, dan untuk Dequeue, cukup "geser"kan Head menjadi elemen selanjutnya.

Operasi pada Queue

- enqueue() : menambahkan data ke dalam queue.
- dequeue() : mengeluarkan data dari queue.
- peek() : mengambil data dari queue tanpa menghapusnya.
- isEmpty() : mengecek apakah queue kosong atau tidak.
- isFull() : mengecek apakah queue penuh atau tidak.

- `size()` : menghitung jumlah elemen dalam queue.

B. GUIDED

Source code

```
#include <iostream>

using namespace std;

const int maksimalQueue = 5;
int front = 0;
int back = 0;
string queueTeller[maksimalQueue];

bool isFull() {
    return back == maksimalQueue;
}

bool isEmpty() {
    return back == 0;
}

void enqueueAntrian(string data) {
    if(isFull()) {
        cout << "Antrian Penuh" << endl;
    } else {
        queueTeller[back] = data;
        back++;
    }
}

void dequeueAntrian() {
    if(isEmpty()) {
        cout << "Antrian kosong" << endl;
    } else {
        for(int i = 0; i < back; i++) {
            queueTeller[i] = queueTeller[i + 1];
        }
        queueTeller[back - 1] = "";
        back--;
    }
}

int countQueue() {
```

```

        return back;
    }

    void clearQueue() {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = "";
        }
    }

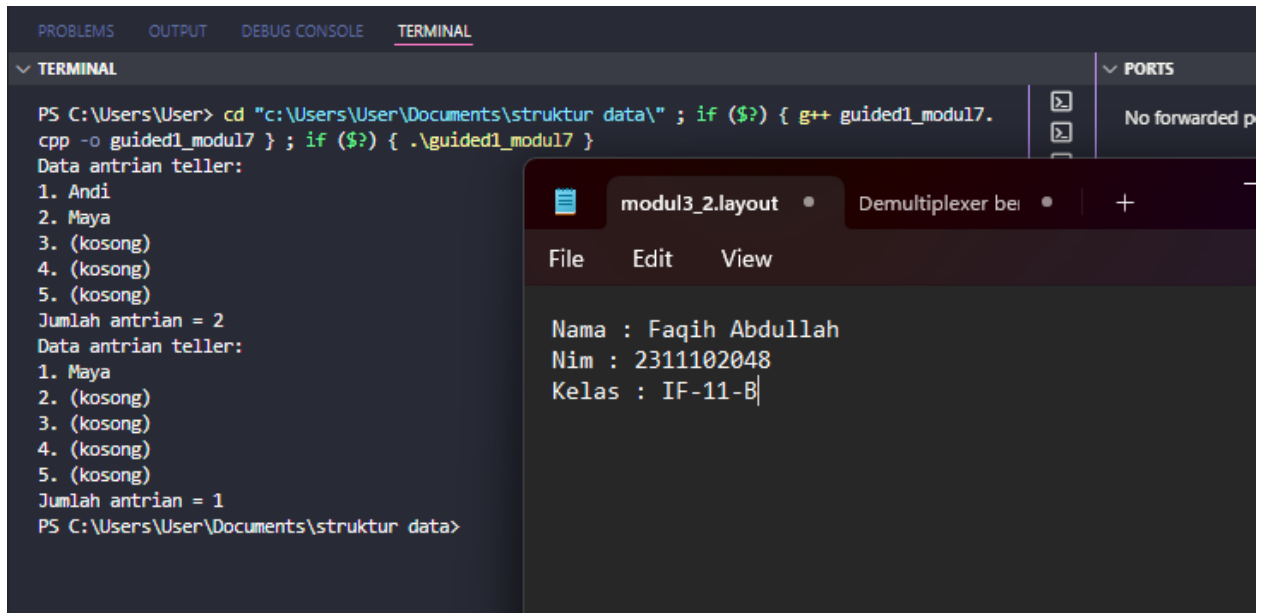
    void viewQueue() {
        cout << "Data antrian teller: " << endl;
        for (int i=0; i < maksimalQueue; i++) {
            if(queueTeller[i] != "") {
                cout << i + 1 << ". "<< queueTeller[i] << endl;
            }else {
                cout << i+1 << ". (kosong)"<< endl;
            }
        }
    }
}

int main() {
    enqueueAntrian("Andi");
    enqueueAntrian("Maya");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    return 0;
}

```

Screenshot Output



```
PS C:\Users\User> cd "c:\Users\User\Documents\struktur data\" ; if ($?) { g++ guided1_modul7.
cpp -o guided1_modul7 } ; if ($?) { .\guided1_modul7 }
Data antrian teller:
1. Andi
2. Maya
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 2
Data antrian teller:
1. Maya
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 1
PS C:\Users\User\Documents\struktur data>
```

```
modul3_2.layout • Demultiplexer bei • +
File Edit View
Nama : Faqih Abdullah
Nim : 2311102048
Kelas : IF-11-B|
```

Deskripsi program

Program di atas merupakan implementasi dari antrian (queue) menggunakan array di bahasa C++. Program ini mendemonstrasikan berbagai operasi dasar pada struktur data antrian seperti menambahkan elemen ke antrian (enqueue), menghapus elemen dari antrian (dequeue), dan menampilkan isi antrian.

C. UNGUIDED

1. ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list

Source Code

```
#include <iostream>

using namespace std;

// Struktur Node untuk linked list
struct Node {
    string data;
    Node* next;
};

// Struktur Queue menggunakan linked list
struct Queue {
    Node* depan;
    Node* belakang;
};
```

```

// Fungsi untuk membuat node baru
Node* buatNode(string data) {
    Node* nodeBaru = new Node();
    nodeBaru->data = data;
    nodeBaru->next = nullptr;
    return nodeBaru;
}

// Fungsi untuk menginisialisasi queue kosong
void inisialisasiQueue(Queue &q) {
    q.depan = nullptr;
    q.belakang = nullptr;
}

// Fungsi untuk memeriksa apakah queue kosong
bool isEmpty(Queue q) {
    return q.depan == nullptr;
}

// Fungsi untuk menambahkan elemen ke queue
void enqueueAntrian(Queue &q, string data) {
    Node* nodeBaru = buatNode(data);
    if (isEmpty(q)) {
        q.depan = nodeBaru;
        q.belakang = nodeBaru;
    } else {
        q.belakang->next = nodeBaru;
        q.belakang = nodeBaru;
    }
}

// Fungsi untuk menghapus elemen dari queue
void dequeueAntrian(Queue &q) {
    if (isEmpty(q)) {
        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = q.depan;
        q.depan = q.depan->next;
        delete temp;
        if (q.depan == nullptr) {
            q.belakang = nullptr;
        }
    }
}

// Fungsi untuk menghitung elemen dalam queue

```

```

int countQueue(Queue q) {
    int count = 0;
    Node* current = q.depan;
    while (current != nullptr) {
        count++;
        current = current->next;
    }
    return count;
}

// Fungsi untuk mengosongkan queue
void clearQueue(Queue &q) {
    while (!isEmpty(q)) {
        dequeueAntrian(q);
    }
}

// Fungsi untuk menampilkan elemen-elemen dalam queue
void viewQueue(Queue q) {
    cout << "Data antrian teller: " << endl;
    Node* current = q.depan;
    int index = 1;
    while (current != nullptr) {
        cout << index << ". " << current->data << endl;
        current = current->next;
        index++;
    }
    if (index == 1) {
        cout << "Antrian kosong" << endl;
    }
}

int main() {
    Queue q;
    inisialisasiQueue(q);

    enqueueAntrian(q, "Andi");
    enqueueAntrian(q, "Maya");
    viewQueue(q);
    cout << "Jumlah antrian = " << countQueue(q) << endl;

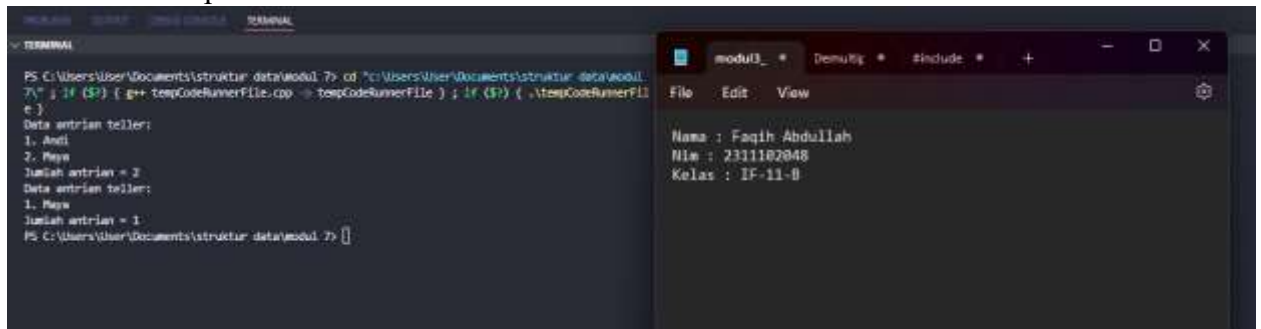
    dequeueAntrian(q);
    viewQueue(q);
    cout << "Jumlah antrian = " << countQueue(q) << endl;

    return 0;
}

```

```
}
```

Screenshot Output



```
PS C:\Users\User\Documents\struktur data\modul 7> cd "C:\Users\User\Documents\struktur data\modul 7" & if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } & if ($?) { .\tempCodeRunnerFile.exe }
Data antrian teller:
1. Andi
2. Maya
Jumlah antrian = 2
Data antrian teller:
1. Maya
Jumlah antrian = 1
PS C:\Users\User\Documents\struktur data\modul 7>
```

```
modul3_ *  DemuRig *  #include *  +
File Edit View
Nama : Faqih Abdullah
Nilai : 2311182848
Kelas : IF-11-B
```

Deskripsi Program

Program diatas adalah pengubahan code dari guided yang diberikan asprak. Dimana penyimpanan data awalnya menggunakan array sekarang menggunakan linked list
Penjelasan lebih detail sebagai berikut:

Struktur Node digunakan untuk merepresentasikan setiap elemen dalam linked list. Struktur Queue memiliki pointer ke elemen depan (front) dan belakang (back) dari antrian.

- Fungsi *createNode* membuat node baru dengan data yang diberikan.
- Fungsi *initializeQueue* menginisialisasi antrian kosong.
- Fungsi *isEmpty* memeriksa apakah antrian kosong.
- Fungsi *enqueueAntrian* menambahkan elemen baru ke belakang antrian.
- Fungsi *dequeueAntrian* menghapus elemen dari depan antrian.
- Fungsi *countQueue* menghitung jumlah elemen dalam antrian.
- Fungsi *clearQueue* mengosongkan antrian.
- Fungsi *viewQueue* menampilkan elemen-elemen dalam antrian.
- Program utama (main) menguji fungsi-fungsi tersebut dengan menambahkan, menampilkan, dan menghapus elemen dalam antrian.

Unguided

2. dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM mahasiswa

Source code

```
#include <iostream>

using namespace std;

// Struktur Node untuk linked list dengan atribut Nama dan NIM
struct Node {
```



```

    string nama;
    string nim;
    Node* next;
};

// Struktur Queue menggunakan linked list
struct Queue {
    Node* depan;
    Node* belakang;
};

// Fungsi untuk membuat node baru
Node* buatNode(string nama, string nim) {
    Node* nodeBaru = new Node();
    nodeBaru->nama = nama;
    nodeBaru->nim = nim;
    nodeBaru->next = nullptr;
    return nodeBaru;
}

// Fungsi untuk menginisialisasi queue kosong
void inisialisasiQueue(Queue &q) {
    q.depan = nullptr;
    q.belakang = nullptr;
}

// Fungsi untuk memeriksa apakah queue kosong
bool isEmpty(Queue q) {
    return q.depan == nullptr;
}

// Fungsi untuk menambahkan elemen ke queue
void enqueueAntrian(Queue &q, string nama, string nim) {
    Node* nodeBaru = buatNode(nama, nim);
    if (isEmpty(q)) {
        q.depan = nodeBaru;
        q.belakang = nodeBaru;
    } else {
        q.belakang->next = nodeBaru;
        q.belakang = nodeBaru;
    }
}

// Fungsi untuk menghapus elemen dari queue
void dequeueAntrian(Queue &q) {
    if (isEmpty(q)) {

```

```

        cout << "Antrian kosong" << endl;
    } else {
        Node* temp = q.depan;
        q.depan = q.depan->next;
        delete temp;
        if (q.depan == nullptr) {
            q.belakang = nullptr;
        }
    }
}

// Fungsi untuk menghitung elemen dalam queue
int countQueue(Queue q) {
    int count = 0;
    Node* current = q.depan;
    while (current != nullptr) {
        count++;
        current = current->next;
    }
    return count;
}

// Fungsi untuk mengosongkan queue
void clearQueue(Queue &q) {
    while (!isEmpty(q)) {
        dequeueAntrian(q);
    }
}

// Fungsi untuk menampilkan elemen-elemen dalam queue
void viewQueue(Queue q) {
    cout << "Data antrian mahasiswa: " << endl;
    Node* current = q.depan;
    int index = 1;
    while (current != nullptr) {
        cout << index << ". Nama: " << current->nama << ", NIM: " <<
current->nim << endl;
        current = current->next;
        index++;
    }
    if (index == 1) {
        cout << "Antrian kosong" << endl;
    }
}

int main() {

```

```

Queue q;
inisialisasiQueue(q);

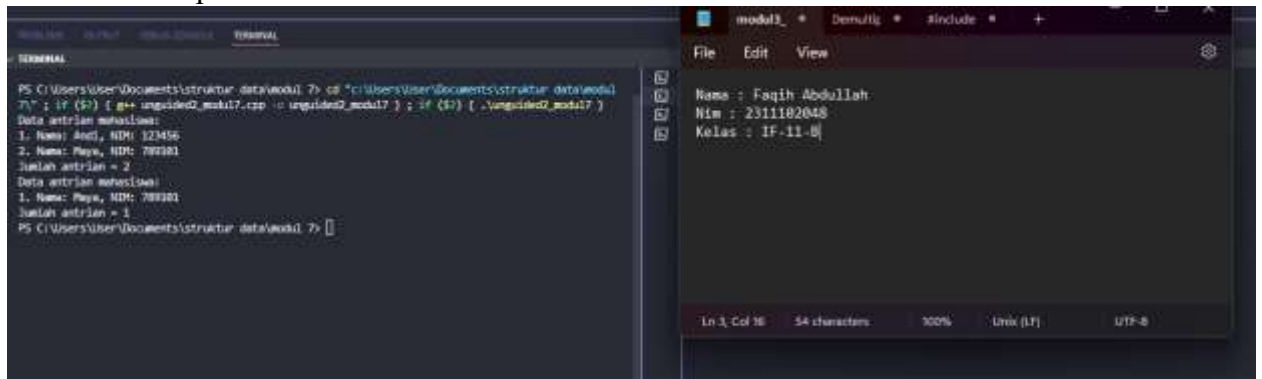
enqueueAntrian(q, "Andi", "123456");
enqueueAntrian(q, "Maya", "789101");
viewQueue(q);
cout << "Jumlah antrian = " << countQueue(q) << endl;

dequeueAntrian(q);
viewQueue(q);
cout << "Jumlah antrian = " << countQueue(q) << endl;

return 0;
}

```

Screenshot output



Deskripsi Program

Program di atas adalah implementasi antrian (queue) menggunakan struktur data linked list dalam bahasa C++. Antrian ini digunakan untuk menyimpan data mahasiswa dengan atribut nama dan NIM (Nomor Induk Mahasiswa).

Penjelasan

Struktur node :

- Mendefinisikan elemen dalam linked list yang terdiri dari dua atribut yaitu 'nama' dan 'nim' untuk menyimpan nama dan NiM mahasiswa.
- Node juga memiliki pointer 'next' yang menunjuk ke node berikutnya dalam linked list.

Struktur queue:

- Struktur queue mendefinisikan antrian yang memiliki dua pointer : 'depan' untuk menunjuk ke elemen pertama dan 'belakang' untuk menunjuk ke elemen terakhir dalam antrian.

Fungsi 'buatNode'

- Fungsi ini membuat node baru dengan atribut 'nama' dan 'nim' yang diberikan, dan menginisialisasikan pointer 'next' ke 'nullptr'.

Fungsi 'inisialisasiQueue' :

- Fungsi ini menginisialisasi antrian kosong dengan mengatur pointer 'depan' dan 'belakang' ke 'nullptr'.

Fungsi 'isEmpty':

- Fungsi ini memeriksa apakah antrian kosong dengan memeriksa apakah pointer 'depan' adalah 'nullptr'.

Fungsi 'enqueueAntrian':

- Fungsi ini menambahkan elemen baru (node) ke antrian. Jika antrian kosong, elemen baru menjadi elemen pertama. Jika tidak, elemen baru ditambahkan di belakang antrian.

Fungsi 'dequeueAntrian':

- Fungsi ini menghapus elemen dari depan antrian. Jika antrian kosong, fungsi mencetak pesan "antrian kosong". Jika tidak, elemen pertama dihapus dan pointer 'depan' di pindahkan ke elemen berikutnya.

Fungsi 'counQueue':

- Fungsi ini menghitung jumlah elemen dalam antrian dengan mengiterasi melalui linked list dari 'depan' ke 'belakang'.

Fungsi 'clearQueue':

- Fungsi ini mengosongkan antrian dengan menghapus semua elemen satu persatu sampai antrian kosong.

Fungsi 'viewQueue':

- Fungsi ini menampilkan semua elemen dalam antrian. Setiap elemen ditampilkan dengan nomor urut, nama, dan NIM. Jika antrian kosong, fungsi ini mencetak pesan "antrian kosong".

Fungsi 'main':

- Fungsi ini menguji semua fungsi diatas dengan scenario sebagai berikut:
 - Membuat antrian kosong
 - Menambahkan dua mahasiswa ke antrian
 - Menampilkan isi antrian dan jumlah elemen dalam antrian
 - Menghapus satu elemen dari antrian
 - Menampilkan isi antrian dan jumlah elemen dalam antrian setelah penghapusan

DAFTAR PUSTAKA

praktikum, a. (2024). modul 7 Queue. *modul 7 Queue*, 1-5.