

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



Disusun Oleh :

Anisah Syifa Mustika Riyanto
2311102080

Dosen

Wahyu Andi Saputra, S.Pd., M.Eng.

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. Dasar Teori

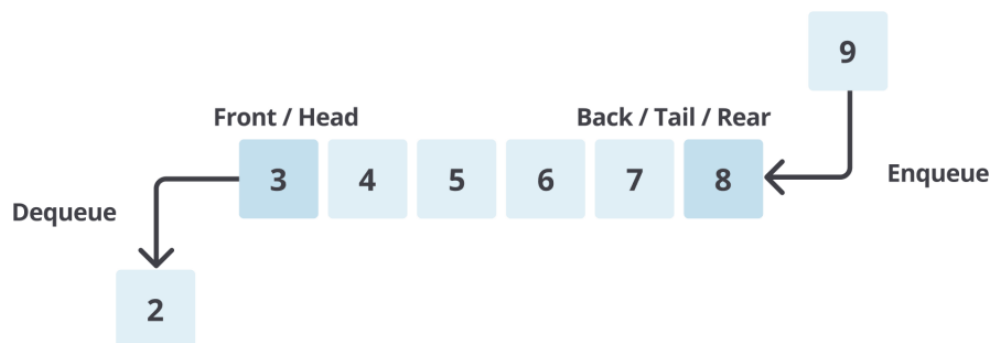
Pengertian Queue

Queue atau dalam bahasa Indonesia yang berarti antrian adalah struktur data yang menyusun elemen-elemen data dalam urutan linier. Prinsip dasar dari struktur data ini adalah “First In, First Out” (FIFO) yang berarti elemen data yang pertama dimasukkan ke dalam antrian akan menjadi yang pertama pula untuk dikeluarkan.

Prinsip FIFO pada Queue

Caranya bekerja adalah seperti jejeran orang yang sedang menunggu antrian di supermarket di mana orang pertama yang datang adalah yang pertama dilayani (First In, First Out). Pada struktur data ini, urutan pertama (data yang akan dikeluarkan) disebut Front atau Head. Sebaliknya, data pada urutan terakhir (data yang baru saja ditambahkan) disebut Back, Rear, atau Tail. Proses untuk menambahkan data pada antrian disebut dengan Enqueue, sedangkan proses untuk menghapus data dari antrian disebut dengan Dequeue.

Queue Data Structure



Fungsi Queue

Queue memiliki peran yang penting dalam berbagai aplikasi dan algoritma. Salah satu fungsi utamanya adalah mengatur dan mengelola antrian tugas atau operasi secara efisien. Dalam sistem komputasi, ia digunakan untuk menangani tugas-tugas seperti penjadwalan proses, antrian pesan, dan manajemen sumber daya.

Jenis-jenis Queue

Jenis struktur data ini dapat diklasifikasikan berdasarkan cara implementasinya, maupun berdasarkan penggunaannya. Di antara jenis-jenis tersebut adalah sebagai berikut.

1. Berdasarkan Implementasinya

- Linear/Simple Queue: Elemen-elemen data disusun dalam barisan linear dan penambahan serta penghapusan elemen hanya terjadi pada dua ujung barisan tersebut.
- Circular Queue: Mirip dengan jenis linear, tetapi ujung-ujung barisan terhubung satu sama lain, menciptakan struktur antrian yang berputar.

2. Berdasarkan Penggunaan

- Priority Queue: Setiap elemen memiliki prioritas tertentu. Elemen dengan prioritas tertinggi akan diambil terlebih dahulu.
- Double-ended Queue (Deque): Elemen dapat ditambahkan atau dihapus dari kedua ujung antrian.

Keuntungan dan Keterbatasan

Meskipun struktur data queue memiliki banyak kegunaan, kamu juga harus mengetahui beberapa keuntungan dan keterbatasan yang perlu diperhatikan sebelum menggunakannya.

Keuntungan

- Data berjumlah besar dapat dikelola dengan mudah dan efisien.
- Proses *insert* dan *delete* data dapat dilakukan dengan mudah karena mengikuti aturan *first in first out* (FIFO).
- Efisien dalam menangani tugas berdasarkan urutan kedatangan.

Keterbatasan

- Tidak efisien untuk pencarian elemen tertentu dalam antrian.
- Memerlukan alokasi memori yang cukup untuk menyimpan antrian.

B. Guided

Source code:

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

const int maksimalQueue = 5; //Batas maksimal antrian
int front = 0;                //Indeks antrian awal
int back = 0;                 //Indeks antrian akhir
string queueTeller[maksimalQueue]; //Array untuk menyimpan elemen antrian

// Fungsi untuk memeriksa apakah antrian penuh
bool isFull(){
    return back == maksimalQueue;
}

//Fungsi untuk memeriksa apakah antrian kosong
bool isEmpty(){
    return back == 0;
}

// Fungsi untuk menambahkan elemen ke antrian
void enqueueAntrian(string data){
    if (isFull()) {
```

```

        cout << "Antrian penuh" << endl;
    }else {
        queueTeller[back]= data;
        back++;
    }
}

// Fungsi untuk menghapus elemen dari antrian
void dequeueAntrian(){
    if (isEmpty()){
        cout << "Antrian kosong" << endl;
    }else {
        for (int i=0; i< back - 1; i++){
            queueTeller[i] = queueTeller[i+1];
        }
        queueTeller [back-1]= ""; // Membersihkan data terakhir
        back--;
    }
}

// Fungsi untuk menghitung jumlah elemen dalam antrian
int countQueue(){
    return back;
}

// Fungsi untuk mengosongkan semua elemen dalam antrian
void clearQueue(){
    for (int i = 0; i < back; i++){
        queueTeller[i] = "";
    }
    back = 0;
    front = 0;
}

// Fungsi untuk menampilkan semua elemen dalam antrian
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");
    enqueueAntrian("Anisah");
    enqueueAntrian("Syifa");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;

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

    dequeueAntrian();
    viewQueue();
    cout << "Jumlah Antrian = " << countQueue() << endl;
}

```

```

clearQueue();
viewQueue();
cout << "Jumlah Antrian = " << countQueue() << endl;

return 0;
}

```

Screenshots Output:

```

PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC> & 'c:\Users\hp151\.vscode\extensions\ms-vscode.cp
ptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-pjufmky4.bme
' '--stdout=Microsoft-MIEngine-Out-z14jgfnh.42i' '--stderr=Microsoft-MIEngine-Error-xkwlfbyk.fjv' '--pid=Micros
oft-MIEngine-Pid-fl0rmjog.0ie' '--dbgExe=C:\Program Files\CodeBlocks\MinGW\bin\gdb.exe' '--interpreter=mi'
Data antrian teller:
1. Andi
2. Maya
3. Anisah
4. Syifa
5. (kosong)
Jumlah antrian = 4
Data antrian teller:
1. Maya
2. Anisah
3. Syifa
4. (kosong)
5. (kosong)
Jumlah Antrian = 3
Data antrian teller:
1. Anisah
2. Syifa
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah Antrian = 2
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah Antrian = 0
PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC>

```

(maaf notenya salah, bukan nama. Tapi di output program udah ada nama Anisah Syifa)

Deskripsi:

Program menggunakan queue untuk mengatur garis teller. Program menggunakan array string untuk menyimpan nama nasabah yang mengantri. Program ini termasuk fungsi untuk memeriksa apakah antrian penuh atau kosong, menambahkan pelanggan ke akhir antrian (`enqueueAntrian`), menghapus pelanggan dari awal antrian (`dequeueAntrian`), menghitung jumlah pelanggan di antrian (`countQueue`), mengosongkan semua pelanggan di antrian (`clearQueue`), dan menampilkan status antrian saat ini (`viewQueue`). Program yang dijalankan menambahkan empat nasabah, menampilkan antrian, menghapus dua antrian, menampilkan antrian yang diperbarui, mengosongkan antrian, dan menampilkan antrian yang kosong.

C. Unguided

1. Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list
2. Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

Source code:

```
// 1. Ubahlah penerapan queue pada guided dari array menjadi linked
list, buatlah konsep antri dengan atribut Nama mahasiswa dan NIM

#include <iostream>

using namespace std;
struct Node
{
    string namaMhs;
    string nimMhs;
    Node *next;
};
class Queue
{
private:
    Node *front;
    Node *back;

public:
    Queue()
    {
        front = nullptr;
        back = nullptr;
    }
    bool isEmpty() { return front == nullptr; }
    void enqueue(string namaMhs, string nimMhs)
    {
        Node *newNode = new Node();
        newNode->namaMhs = namaMhs;
        newNode->nimMhs = nimMhs;
        newNode->next = nullptr;
        if (isEmpty())
        {
            front = newNode;
            back = newNode;
        }
        else
        {
            back->next = newNode;
            back = newNode;
        }
    }
    void dequeue()
    {
        if (isEmpty())
        {
            cout << "Antrean kosong" << endl;
        }
        else
        {

```

```

        Node *temp = front;
        front = front->next;
        delete temp;
    }
}
int countQueue()
{
    int count = 0;
    Node *temp = front;
    while (temp != nullptr)
    {
        count++;
        temp = temp->next;
    }
    return count;
}
void deleteQueue()
{
    while (!isEmpty())
    {
        dequeue();
    }
}
void printQueue()
{
    cout << "Data antrian Mahasiswa: " << endl;
    Node *temp = front;
    int pos = 1;
    while (temp != nullptr)
    {
        cout << pos << ". Nama: " << temp->namaMhs << "|| NIM: "
        << temp->nimMhs << endl;
        temp = temp->next;
        pos++;
    }
}
};

int main()
{
    Queue queue;
    int pilihan;
    do
    {
        cout << "-----" <<
endl;
        cout << "    Data Antre Hats Coffee Khusus Mahasiswa    " <<
endl;
        cout << "-----" <<
endl;
        cout << "1. Tambah antrean" << endl;
        cout << "2. Hapus antrean (selesai)" << endl;
        cout << "3. Hapus antrean (semua)" << endl;
        cout << "4. Exit" << endl;
        cout << "-----\n" <<
endl;
        cout << "Masukkan Pilihan: ";
        cin >> pilihan;
        switch (pilihan)
        {
            case 1:

```

```

        {
            string namaMhs, nimMhs;
            cout << "Nama Mahasiswa : ";
            cin.ignore();
            getline(cin, namaMhs);
            cout << "NIM Mahasiswa : ";
            cin >> nimMhs;
            queue.enqueue(namaMhs, nimMhs);
            cout << "Anda telah ditambahkan dalam antrean 'Hats
Coffee'" << endl<<endl;
            queue.printQueue();
            break;
        }
        case 2:
        {
            if (queue.isEmpty())
            {
                cout << "Antrean kosong" << endl;
            }
            else
            {
                queue.dequeue();
                cout << "Satu antrean dihapus" << endl<<endl;
            }
            queue.printQueue();
            break;
        }
        case 3:
        {
            if (queue.isEmpty())
            {
                cout << "Antrean kosong" << endl;
            }
            else
            {
                queue.deleteQueue();
                cout << "Data seluruh antrean telah dihapus" << endl;
            }
            break;
        }
        case 4:
        {
            cout << "Terima kasih telah membeli 'Hats Coffee' " <<
endl;
            break;
        }
        default:
        {
            cout << "Pilihan tidak valid" << endl;
            break;
        }
        }
        cout << endl;
    } while (pilihan != 4);
    return 0;
}

```


Screenshots Output:

```
-----
Data Antre Hats Coffee Khusus Mahasiswa
-----
1. Tambah antrean
2. Hapus antrean (selesai)
3. Hapus antrean (semua)
4. Exit
-----

Masukkan Pilihan: 1
Nama Mahasiswa : IF 11 B
NIM Mahasiswa : 333
Anda telah ditambahkan dalam antrean 'Hats Coffee'

Data antrian Mahasiswa:
1. Nama: Anisah|| NIM: 111
2. Nama: Syifa|| NIM: 222
3. Nama: IF 11 B|| NIM: 333
-----

Data Antre Hats Coffee Khusus Mahasiswa
-----
1. Tambah antrean
2. Hapus antrean (selesai)
3. Hapus antrean (semua)
4. Exit
-----

Masukkan Pilihan: 2
Satu antrean dihapus

Data antrian Mahasiswa:
1. Nama: Syifa|| NIM: 222
2. Nama: IF 11 B|| NIM: 333
-----

Data Antre Hats Coffee Khusus Mahasiswa
-----
1. Tambah antrean
2. Hapus antrean (selesai)
3. Hapus antrean (semua)
4. Exit
-----

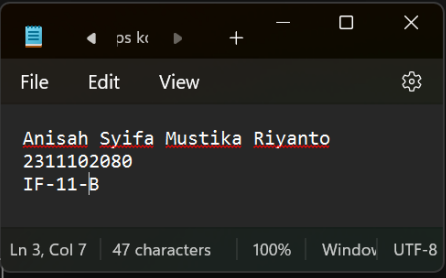
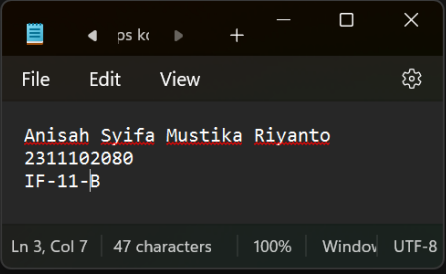
Masukkan Pilihan: 3
Data seluruh antrean telah dihapus

-----

Data Antre Hats Coffee Khusus Mahasiswa
-----
1. Tambah antrean
2. Hapus antrean (selesai)
3. Hapus antrean (semua)
4. Exit
-----

Masukkan Pilihan: 4
Terima kasih telah membeli 'Hats Coffee'

PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC\Modul 7>
```



Deskripsi:

Program ini menerapkan queue dengan linked list. Program merupakan sistem antre khusus untuk mahasiswa yang ingin membeli "Hats Coffee". Pengguna harus memasukkan nama dan nim pembeli/ mahasiswa untuk menambah antrean. Pengguna dapat memilih untuk menambahkan antri, menghapus antri yang terakhir ditambahkan, menghapus semua antri, atau keluar dari program menggunakan fungsi yang telah dibuat, yakni void enqueue(), void dequeue(), void deleteQueue(), dan void printQueue().

D. Kesimpulan

Berikut adalah kesimpulan mengenai materi queue:

1. Queue adalah struktur data yang mengatur elemen-elemen data dalam urutan linier, dengan prinsip operasi "First In, First Out" (FIFO), mirip dengan antrean di kehidupan sehari-hari.
2. Queue memiliki kelebihan seperti efisiensi dalam pengelolaan data besar dan operasi penyisipan/penghapusan yang mudah, namun juga memiliki kekurangan seperti tidak efisien untuk pencarian elemen tertentu
3. Queue dapat dikelompokkan menjadi jenis-jenis seperti Linear/Simple Queue, Circular Queue, Priority Queue, dan Double-ended Queue (Deque) berdasarkan cara implementasinya dan penggunaannya.
4. Operasi enqueue menambahkan elemen ke akhir queue, sedangkan operasi dequeue menghapus elemen dari awal queue.

E. Referensi

Adlaimi, N. (2019). Struktur Data Majemuk (Queue).

Dharmayanti, D. (2011). Queue.

Mukharil Bachtiar, A. (2012). Queue (pelengkap).

Mukharil Bachtiar, A. (2010). Bab IX-Queue.

Maulana, Riski. (2023). Struktur Data Queue: Pengertian, Fungsi, dan Jenisnya. Diakses 22 Mei 2024 dari <https://www.dicoding.com/blog/struktur-data-queue-pengertian-fungsi-dan-jenisnya/>