

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL VII
QUEUE**



Disusun Oleh :

NAMA : ANISA YASAROH

NIM : 2311102063

Dosen :

WAHYU ANDI SAPUTRA, S.Pd., M.Eng.

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

A. DASAR TEORI

a. 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.

Berikut ini adalah ilustrasi dari queue



Pada gambar di atas, karena elemen 1 ditambahkan ke antrian lebih dulu daripada 2, maka 1 adalah elemen yang pertama dihapus dari antrian. Hal ini mengikuti aturan operasi FIFO. Dalam istilah pemrograman, menempatkan item dalam struktur data queue disebut enqueue, sedangkan operasi menghapus item dari queue disebut dequeue. Struktur data queue umumnya digunakan untuk mengelola thread dalam multithreading dan menerapkan sistem antrian prioritas pada program komputer.

b. 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.

c. 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**.

a) 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

b) 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

1. GUIDED

1. Guided 1 – Hash Table

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

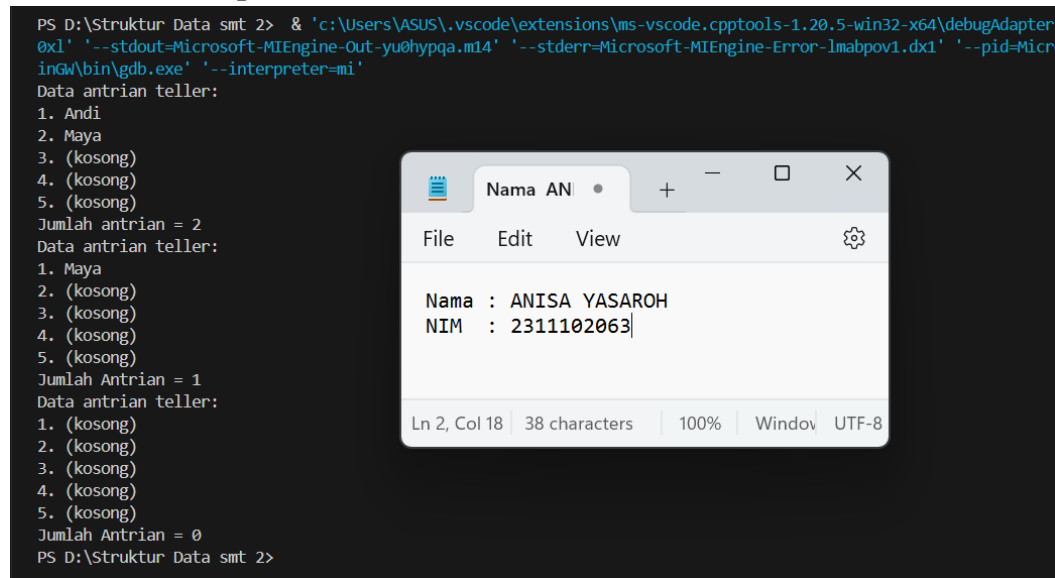
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah Antrian = " << countQueue() << endl;

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

    return 0;
}

```

Screenshoot Output



```
PS D:\Struktur Data smt 2> & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapter\
0x1' '--stdout=Microsoft-MIEngine-Out-yu0hypqa.m14' '--stderr=Microsoft-MIEngine-Error-lmabpov1.d1' '--pid=Micro
inGW\bin\gdb.exe' '--interpreter=mi'
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
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah Antrian = 0
PS D:\Struktur Data smt 2>
```

The screenshot shows a terminal window with the output of a queue program. The program starts with an empty queue and adds 'Andi' and 'Maya'. It then displays the queue state and the count of elements. A small application window titled 'Nama AN' is overlaid on the terminal, showing the input 'ANISA YASAROH' and '2311102063'.

Deskripsi Program

Program di atas yaitu implementasi antrian (queue) menggunakan array, dengan kapasitas maksimal lima elemen. Program mencakup fungsi-fungsi untuk memeriksa apakah antrian penuh atau kosong, menambahkan elemen ('enqueue'), menghapus elemen ('dequeue'), menghitung jumlah elemen dalam antrian, mengosongkan seluruh antrian, dan menampilkan isi antrian. Pada fungsi 'main', program menambahkan elemen ke antrian, menampilkan antrian, menghitung dan menampilkan jumlah elemen, menghapus satu elemen, dan mengosongkan antrian, sambil menampilkan hasil setiap operasi. Antrian diimplementasikan dengan cara menggeser elemen-elemen ketika elemen terdepan dihapus.

2. UNGUIDED

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

Source Code

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

const int maximalQueue = 5;
int length = 0;

struct Node
{
    string data_63;
    Node *next;
};
```

```
Node *head;
Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

bool isFull()
{
    return (length == maximalQueue);
}

bool isEmpty()
{
    return head == NULL;
}

void enqueueAntrian(string nilai_63)
{
    if (isFull())
    {
        cout << "Antrian Penuh" << endl;
    }
    else
    {
        Node *baru = new Node;
        baru->data_63 = nilai_63;
        baru->next = NULL;
        if (isEmpty())
        {
            head = tail = baru;
        }
        else
        {
            tail->next = baru;
            tail = baru;
        }
        length++;
    }
}

void dequeueAntrian()
{
    if (!isEmpty())
    {
        Node *hapus = head;
```

```

        if (head->next != NULL)
        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "Tidak ada antrian" << endl;
    }
}

void clearQueue()
{
    Node *bantu = head;
    while (bantu != NULL)
    {
        Node *hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "Semua antrian dihapus" << endl;
}

void viewQueue()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        int index = 1;
        while (bantu != NULL)
        {
            cout << index << ". " << bantu->data_63 << " "
<< endl;
            bantu = bantu->next;
            index++;
        }
        cout << endl;
    }
    else
    {
        cout << "Tidak ada antrian" << endl;
    }
}

```



```

    }
}

int countQueue()
{
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

int main()
{
    init();
    enqueueAntrian("Andi");
    enqueueAntrian("Maya");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    return 0;
}

```

Screenshoot Output

```

PS D:\Struktur Data smt 2> & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\ft-MIEngine-In-yu0v1qw3.e2q' '--stdout=Microsoft-MIEngine-Out-raztp10i.mrp' '--stderr=Microsoft-MIEngine-Out-raztp10i.mrp' '--dbgExe=C:\Program Files\CodeBlocks\MingW\bin\gdb.exe' '--interpreter=mi'
1. Andi
2. Maya

Jumlah antrian = 2
1. Maya

Jumlah antrian = 1
Semua antrian dihapus
Tidak ada antrian
Jumlah antrian = 0
PS D:\Struktur Data smt 2>

```

VS Code Editor Window: Nama AN

File Edit View

Nama : ANISA YASAROH
NIM : 2311102063

Ln 2, Col 18 | 38 characters | 100% | Window UTF-8

Deskripsi Program

Program di atas yaitu implementasi antrian menggunakan linked list dengan kapasitas maksimal 5 elemen. Fungsi `init()` menginisialisasi antrian, `isFull()` dan `isEmpty()` mengecek kondisi antrian, `enqueueAntrian()` menambahkan elemen baru ke akhir antrian, dan `dequeueAntrian()` menghapus elemen dari depan antrian. Fungsi `clearQueue()` menghapus semua elemen, sementara `viewQueue()` menampilkan semua elemen dalam antrian, dan `countQueue()` menghitung jumlah elemen dalam antrian. Pada fungsi `main()`, program menginisialisasi antrian, menambahkan dua elemen ("Andi" dan "Maya"), menampilkan dan menghitung elemen, menghapus satu elemen, kemudian menampilkan dan menghitung kembali, serta menghapus semua elemen dan menampilkan hasil akhirnya.

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

Source Code

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

const int maximalQueue = 5;
int length = 0;

struct Node
{
    string nama_63;
    string nim_63;
    Node *next;
};

Node *head;
Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

bool isFull()
{
    return (length == maximalQueue);
}

bool isEmpty()
{
    return head == NULL;
}

void enqueueAntrian(string nama_63, string nim_63)
{
    if (isFull())
```

```

    {
        cout << "Antrian penuh" << endl;
    }
    else
    {
        Node *baru = new Node;
        baru->nama_63 = nama_63;
        baru->nim_63 = nim_63;
        baru->next = NULL;
        if (isEmpty())
        {
            head = tail = baru;
        }
        else
        {
            tail->next = baru;
            tail = baru;
        }
        length++;
        cout << endl << "Berhasil Masuk Antrian" << endl;
    }
}

void dequeueAntrian()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        if (head->next != NULL)
        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
        length--;
    }
    else
    {
        cout << "Tidak ada antrian" << endl;
    }
}

void clearQueue()
{
    Node *bantu = head;
    while (bantu != NULL)
    {
        Node *hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
}

```

```

        head = tail = NULL;
        length = 0;
    }

void viewQueue()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        int index = 1;
        while (bantu != NULL)
        {
            cout << index << ". " << bantu->nama_63 << " - "
<< bantu->nim_63 << endl;
            bantu = bantu->next;
            index++;
        }
        cout << endl;
    }
    else
    {
        cout << "Antrian masih kosong" << endl;
    }
}

int countQueue()
{
    return length;
}

int main()
{
    init();
    do
    {
        int choice;
        string nama_63, nim_63;
        cout << "-----" <<
endl;
        cout << "                ANTRIAN MAHASISWA                " <<
endl;
        cout << "-----" <<
endl;
        cout << "1. Tambah Antrian" << endl;
        cout << "2. Keluar Antrian" << endl;
        cout << "3. Jumlah Antrian" << endl;
        cout << "4. Lihat Antrian" << endl;
        cout << "5. Hapus Antrian " << endl;
        cout << "0. Keluar" << endl;
        cout << "Pilih : ";
        cin >> choice;
        cout << endl;
        switch (choice)
        {
            case 1:

```

```

        cout << "Masukkan Nama : ";
        cin.ignore();
        getline(cin, nama_63);
        cout << "Masukkan NIM : ";
        cin >> nim_63;
        enqueueAntrian(nama_63, nim_63);
        break;

    case 2:
        dequeueAntrian();
        cout << "Berhasil keluar" << endl;
        break;

    case 3:
        cout << "Jumlah Antrian : " << countQueue() <<
endl;
        break;

    case 4:
        viewQueue();
        break;

    case 5:
        clearQueue();
        cout << "Data berhasil dihapus" << endl;
        break;

    case 0:
        cout << "Terima kasih telah menggunakan program
ini" << endl;
        return 0;

    default:
        cout << "Pilihan tidak valid" << endl;
        break;
    }
} while (true);

return 0;
}

```

Screenshoot Program

```
yyp.t41' '--dbgExe=C:\Program Files\CodeBlocks\MingW\bin\gdb.exe' '--interpreter=mi'
```

ANTRIAN MAHASISWA

1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 1

Masukkan Nama : Anisa Yasaroh
Masukkan NIM : 2311102063

Berhasil Masuk Antrian

ANTRIAN MAHASISWA

1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 1

Masukkan Nama : Fitri Kusumaningtyas
Masukkan NIM : 2311102068

Berhasil Masuk Antrian

ANTRIAN MAHASISWA

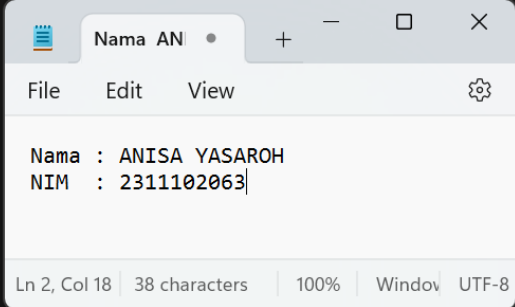
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 1

Masukkan Nama : Trie Nabila Farhah
Masukkan NIM : 2311102071

Berhasil Masuk Antrian

ANTRIAN MAHASISWA

1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 4

1. Anisa Yasaroh - 2311102063
2. Fitri Kusumaningtyas - 2311102068
3. Trie Nabila Farhah - 2311102071

```
-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 2
```

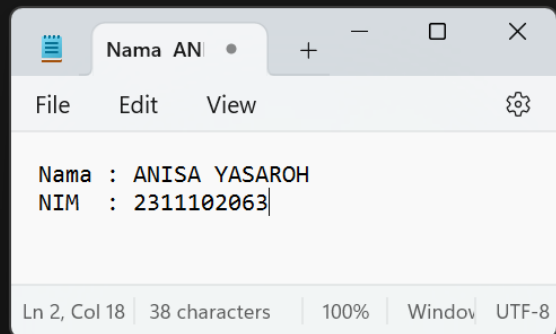
Berhasil keluar

```
-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 3
```

Jumlah Antrian : 2

```
-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 4
```

```
1. Fitri Kusumaningtyas - 2311102068
2. Trie Nabila Farhah - 2311102071
```



```
-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 5
```

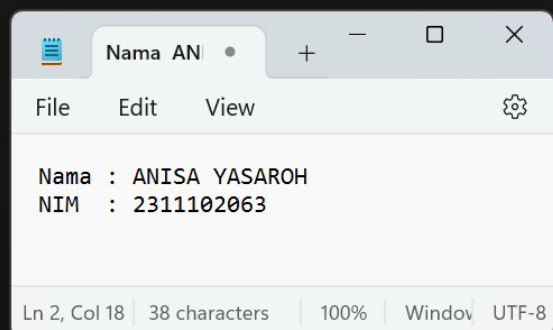
Data berhasil dihapus

```
-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 3
```

Jumlah Antrian : 0

```
-----
ANTRIAN MAHASISWA
-----
1. Tambah Antrian
2. Keluar Antrian
3. Jumlah Antrian
4. Lihat Antrian
5. Hapus Antrian
0. Keluar
Pilih : 0
```

Terima kasih telah menggunakan program ini
PS D:\Struktur Data smt 2> █



Deskripsi Program

Program di atas yaitu implementasi antrian mahasiswa menggunakan linked list dengan batas maksimal 5 elemen. Program mendefinisikan struktur `Node` yang menyimpan nama (`nama_63`) dan NIM (`nim_63`) mahasiswa serta pointer ke node berikutnya. Fungsi `init()` menginisialisasi antrian, `isFull()` dan `isEmpty()` mengecek status antrian, `enqueueAntrian()` menambahkan elemen baru ke antrian jika belum penuh, dan `dequeueAntrian()` menghapus elemen dari depan antrian jika tidak kosong. Fungsi `clearQueue()` menghapus semua elemen dalam antrian, `viewQueue()` menampilkan semua elemen dalam antrian, dan `countQueue()` menghitung jumlah elemen. Fungsi `main()` menyediakan antarmuka menu untuk pengguna dengan opsi untuk menambah, menghapus, melihat, dan menghitung elemen dalam antrian, serta menghapus semua elemen atau keluar dari program. Pengguna dapat memilih tindakan yang diinginkan, dan program akan memproses sesuai pilihan tersebut dalam sebuah loop hingga pengguna memilih untuk keluar.

3. KESIMPULAN

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa :

1. Queue (antrian) adalah salah satu list linier dari struktur data yang beroperasi dengan cara First In First Out (FIFO) yaitu elemen pertama yang masuk merupakan elemen yang pertama keluar.
2. Sebuah queue dalam program setidaknya harus mengandung tiga variabel, yakni: head untuk penanda bagian depan antrian, tail untuk penanda bagian belakang antrian, dan array data untuk menyimpan data-data yang dimasukkan ke dalam queue tersebut.
3. Queue dilakukan dengan cara penyisipan di satu ujung, sedang penghapusan di ujung lain. Ujung penyisipan biasa disebut rear/tail, sedang ujung penghapusan disebut front/head.

4. REFERENSI

Adlaimi, N. (2019). STRUKTUR DATA MAJEMUK (QUEUE).

Anita Sindar, R. M. S. (2019). *Struktur Data Dan Algoritma Dengan C++* (Vol. 1). CV. AA. RIZKY.

Erkamim, E., Abdurrohman, I., Yuliyanti, S., Karim, R., Rahman, A., Admira, T. M. A., & Ridwan, A. (2024). *Buku Ajar Algoritma dan Struktur Data*. PT. Sonpedia Publishing Indonesia.

Holle, K. F. H. (2022). Modul praktikum struktur data.

Nugroho, A. S. Algoritma & Struktur Data Algoritma & Struktur Data.