

LAPORAN PRAKTIKUM QUEUE DALAM BAHASA C



Oleh:

Nama : L Hafidl Alkhair
NIM : 2023903430060
Kelas : TRKJ 1.C
Jurusan : Teknologi Informasi dan Komputer
Program Studi : Teknologi Rekayasa Komputer Jaringan
Dosen Pembimbing : Indrawati, SST. MT



**JURUSAN TEKNOLOGI INFORMASI KOMPUTER
PRODI TEKNOLOGI REKAYASA KOMPUTER JARINGAN
POLITEKNIK NEGERI LHOKSEUMAWE
TAHUN AJARAN 2023/2024**

LEMBAR PENGESAHAN

No. Praktikum : 08 /TIK/TRKJ-1C/ Data Structure And Algorithms Practice

Judul : Laporan Pratikum Bahasa C

Nama : L Hafidl Alkhair

NIM : 2023903430060

Kelas : TRKJ-1C

Jurusan : Teknologi Informasi Dan Komputer

Prodi : Teknologi Rekayasa Komputer Jaringan

Tanggal Praktikum : 15 November 2023

Tanggal Penyerahan : 20 November 2023

Buketrata, 20 November 2023
Dosen Pembimbing,

Indrawati, SST.MT
Nip. 19740815 200112 2 001

1.1. PENDAHULUAN

A. Tujuan

1. Mampu menguasai konsep penulisan program Single Linked List(SSL) dalam bahasa C.
2. Mampu menggunakan program Single Linked List(SSL) dalam bahasa C.
3. Mampu menyelesaikan masalah dengan memanfaatkan Single Linked List(SSL) dalam bahasa C.

B. Alat dan Bahan:

- Komputer atau Laptop Dengan Kompiler C (GCC)
- Editor Teks (Misalnya Notepad ++, DEV – C++)
- Buku Panduan atau Materi Pratikum
- Kertas dan Pensil
- Ruang Praktikum

C. Dasar Teori

konsep antrian (queue) dalam bahasa C mencakup pemahaman mendalam tentang suatu struktur data yang dirancang secara khusus untuk penyimpanan dan pengelolaan elemen-elemen data dengan mengaplikasikan prinsip dasar First In First Out (FIFO). Prinsip FIFO, yang merupakan pilar utama dalam paradigma antrian, menjelaskan bahwa elemen data yang pertama kali dimasukkan ke dalam antrian akan diambil pertama kali pula, membentuk suatu sistem urutan yang terstruktur dan konsisten. Dalam implementasinya, struktur data antrian dalam bahasa C menyajikan suatu kerangka kerja yang tidak hanya memfasilitasi penyimpanan dan pengambilan data dengan urutan yang terjamin, tetapi juga memberikan dasar untuk manajemen data yang efisien dan efektif. Penggunaan antrian ini sangat bermanfaat dalam mengelola situasi di mana elemen-elemen data harus diakses sesuai dengan urutan waktu kedatangannya, seperti pada sistem antrean tugas, antrian perintah eksekusi, atau bahkan pada mekanisme komunikasi antar modul. Sebagai

inti dari banyak aplikasi pengolahan data, konsep antrian dalam bahasa C membawa dampak signifikan pada pembangunan sistem yang handal dan terstruktur. Pengintegrasian prinsip FIFO membantu menciptakan alur kerja yang teratur, memudahkan pengelolaan data dalam skenario yang melibatkan berbagai proses, dan menghasilkan sistem yang dapat diandalkan dalam menjaga integritas dan urutan data. Oleh karena itu, pemahaman mendalam terhadap teori antrian dalam bahasa C bukan hanya memberikan wawasan tentang struktur data, tetapi juga memberikan dasar yang kokoh untuk pengembangan program yang efisien, dapat diandalkan, dan sesuai dengan tuntutan kompleksitas pengolahan data dalam berbagai konteks aplikasi.

1.2. URAIAN PRAKTIKUM

1. Implementasi Queue sederhana menggunakan array:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 5
struct Queue {
    int items[MAX_SIZE];
    int front;
    int rear;
};

void initializeQueue(struct Queue *q) {
    q->front = -1;
    q->rear = -1;
}

int isFull(struct Queue *q) {
    return (q->rear == MAX_SIZE - 1);
}

int isEmpty(struct Queue *q) {
    return (q->front == -1 && q->rear == -1);
}

void enqueue(struct Queue *q, int value) {
    if (isFull(q)) {
        printf("Queue penuh, enqueue gagal\n");
    } else {
        if (isEmpty(q)) {
            q->front = q->rear = 0;
        } else {
            q->rear++;
        }
        q->items[q->rear] = value;
        printf("%d berhasil ditambahkan ke dalam queue\n", value);
    }
}

int dequeue(struct Queue *q) {
    int value;
    if (isEmpty(q)) {
        printf("Queue kosong, dequeue gagal\n");
        return -1;
    } else {
        value = q->items[q->front];
        if (q->front == q->rear) {
            initializeQueue(q);
        } else {
            q->front++;
        }
        printf("%d berhasil dihapus dari queue\n", value);
        return value;
    }
}

void display(struct Queue *q) {
    if (isEmpty(q)) {
        printf("Queue kosong\n");
    } else {
        printf("Isi Queue: ");
        for (int i = q->front; i <= q->rear; i++) {
            printf("%d ", q->items[i]);
        }
        printf("\n");
    }
}

int main() {
    struct Queue myQueue;
    initializeQueue(&myQueue);

    enqueue(&myQueue, 10);
    enqueue(&myQueue, 20);
    enqueue(&myQueue, 30);

    display(&myQueue);

    dequeue(&myQueue);

    display(&myQueue);

    return 0;
}
```

Output:

```
D:\TUGAS-BUK-INDRA\PUNYI >
10 berhasil ditambahkan ke dalam queue
20 berhasil ditambahkan ke dalam queue
30 berhasil ditambahkan ke dalam queue
Isi Queue: 10 20 30
10 berhasil dihapus dari queue
Isi Queue: 20 30
=====
```

Analisa:

Program tersebut adalah implementasi queue menggunakan array dalam bahasa C dengan fitur penuh untuk menambahkan, menghapus, dan menampilkan elemen-elemen dalam queue. Berikut adalah analisis program tersebut:

1. Struktur Data Queue:

- Program menggunakan struktur data Queue untuk menyimpan elemen-elemen.
- Struktur Queue memiliki array items sebagai wadah elemen-elemen, serta variabel front dan rear untuk menandai posisi depan dan belakang dari queue.

2. Fungsi-fungsi Utama:

- initializeQueue: Menginisialisasi queue dengan mengatur front dan rear ke -1.
- isFull: Memeriksa apakah queue penuh.
- isEmpty: Memeriksa apakah queue kosong.
- enqueue: Menambahkan elemen ke dalam queue.
- dequeue: Menghapus elemen dari depan queue dan mengembalikan nilainya.
- display: Menampilkan elemen-elemen dalam queue.

3. Implementasi Queue:

- Elemen-elemen ditambahkan ke queue menggunakan fungsi enqueue.
- Elemen-elemen dihapus dari queue menggunakan fungsi dequeue.

- Fungsi display digunakan untuk menampilkan elemen-elemen dalam queue.

4. Fungsi Utama (main):

-
- Membuat queue baru menggunakan initializeQueue.
- Menambahkan beberapa elemen ke dalam queue menggunakan enqueue.
- Menampilkan elemen dalam queue menggunakan display.
- Menghapus elemen dari queue menggunakan dequeue.
- Menampilkan kembali elemen dalam queue menggunakan display.

2. Antrian Prioritas Menggunakan Queue:

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

#define MAX_SIZE 5

struct Queue {
    int items[MAX_SIZE];
    int front;
    int rear;
};

void initializeQueue(struct Queue *q) {
    q->front = -1;
    q->rear = -1;
}

int isFull(struct Queue *q) {
    return (q->rear == MAX_SIZE - 1);
}

int isEmpty(struct Queue *q) {
    return (q->front == -1 && q->rear == -1);
}

void enqueue(struct Queue *q, int value) {
    if (isFull(q)) {
        printf("Queue penuh, enqueue gagal\n");
    } else {
        if (isEmpty(q)) {
            q->front = q->rear = 0;
        } else {
            q->rear++;
        }
        q->items[q->rear] = value;
        printf("%d berhasil ditambahkan ke dalam queue\n", value);
    }
}

int dequeue(struct Queue *q) {
    int value;
    if (isEmpty(q)) {
        printf("Queue kosong, dequeue gagal\n");
        return -1;
    } else {
        int highestPriorityIndex = q->front;
        for (int i = q->front + 1; i <= q->rear; i++) {
            if (q->items[i] < q->items[highestPriorityIndex]) {
                highestPriorityIndex = i;
            }
        }
        value = q->items[highestPriorityIndex];
        // Shift elements to fill the gap
        for (int i = highestPriorityIndex; i < q->rear; i++) {
            q->items[i] = q->items[i + 1];
        }
        q->rear--;
    }

    if (q->front > q->rear) {
        initializeQueue(q);
    }

    printf("%d dengan prioritas tertinggi berhasil dihapus dari queue\n", value);
    return value;
}

void display(struct Queue *q) {
    if (isEmpty(q)) {
        printf("Queue kosong\n");
    } else {
        printf("Isi Queue: ");
        for (int i = q->front; i <= q->rear; i++) {
            printf("%d ", q->items[i]);
        }
        printf("\n");
    }
}

int main() {
    struct Queue myQueue;
    initializeQueue(&myQueue);

    enqueue(&myQueue, 30);
    enqueue(&myQueue, 10);
    enqueue(&myQueue, 50);
    enqueue(&myQueue, 20);

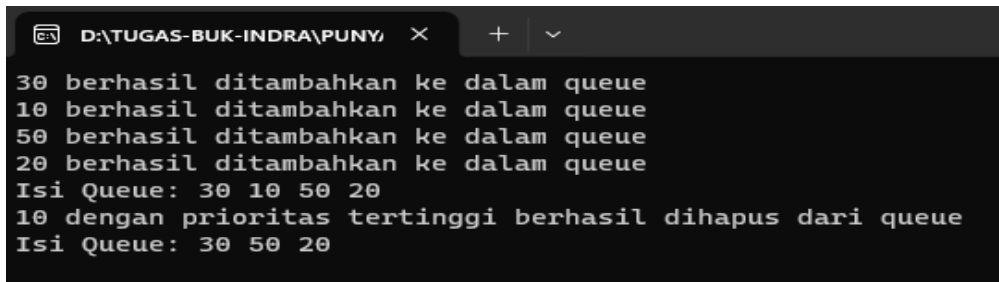
    display(&myQueue);

    dequeue(&myQueue);

    display(&myQueue);

    return 0;
}
```

Output:



```
D:\TUGAS-BUK-INDRA\PUNY
30 berhasil ditambahkan ke dalam queue
10 berhasil ditambahkan ke dalam queue
50 berhasil ditambahkan ke dalam queue
20 berhasil ditambahkan ke dalam queue
Isi Queue: 30 10 50 20
10 dengan prioritas tertinggi berhasil dihapus dari queue
Isi Queue: 30 50 20
```

Analisa:

Program ini adalah implementasi sederhana dari struktur data queue dengan prioritas dalam bahasa C. Queue dengan prioritas mengutamakan penghapusan elemen dengan nilai tertinggi terlebih dahulu. Berikut adalah analisis singkatnya:

1. Struktur Data Queue:

- Program menggunakan struktur data Queue untuk menyimpan elemen-elemen.
- Struktur Queue memiliki array items sebagai wadah elemen-elemen, serta variabel front dan rear untuk menandai posisi depan dan belakang dari queue.

2. Fungsi-fungsi Utama:

- initializeQueue: Menginisialisasi queue dengan mengatur front dan rear ke -1.
- isFull: Memeriksa apakah queue penuh.
- isEmpty: Memeriksa apakah queue kosong.
- enqueue: Menambahkan elemen ke dalam queue dengan prioritas tertinggi.
- dequeue: Menghapus elemen dengan prioritas tertinggi dari queue dan mengembalikan nilainya.
- display: Menampilkan elemen-elemen dalam queue.

3. Implementasi Queue dengan Prioritas:

- Fungsi enqueue menambahkan elemen dengan cara mencari indeks elemen dengan nilai terendah dan menempatkannya di posisi itu.
- Fungsi dequeue mencari elemen dengan nilai tertinggi, menghapusnya,

dan menyesuaikan posisi elemen-elemen lain jika perlu.

4. Fungsi Utama (main):

- Membuat queue baru menggunakan initializeQueue.
- Menambahkan beberapa elemen dengan prioritas ke dalam queue menggunakan enqueue.
- Menampilkan elemen dalam queue menggunakan display.
- Menghapus elemen dengan prioritas tertinggi dari queue menggunakan dequeue.
- Menampilkan kembali elemen dalam queue menggunakan display.

3. Program antrian (queue) yang menangani string sebagai elemen

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_SIZE 5
#define MAX_STRING_LENGTH 50

struct Queue {
    char items[MAX_SIZE][MAX_STRING_LENGTH];
    int front;
    int rear;
};

void initializeQueue(struct Queue *q) {
    q->front = -1;
    q->rear = -1;
}

int isFull(struct Queue *q) {
    return (q->rear == MAX_SIZE - 1);
}

int isEmpty(struct Queue *q) {
    return (q->front == -1 && q->rear == -1);
}

void enqueue(struct Queue *q, const char *value) {
    if (isFull(q)) {
        printf("Queue is full, enqueue failed\n");
    } else {
        if (isEmpty(q)) {
            q->front = q->rear = 0;
        } else {
            q->rear++;
        }
        strcpy(q->items[q->rear], value);
        printf("%s successfully added to the queue\n", value);
    }
}

char* dequeue(struct Queue *q) {
    static char value[MAX_STRING_LENGTH];
    if (isEmpty(q)) {
        printf("Queue is empty, dequeue failed\n");
        return NULL;
    } else {
        strcpy(value, q->items[q->front]);
        if (q->front == q->rear) {
            initializeQueue(q);
        } else {
            q->front++;
        }
        printf("%s successfully removed from the queue\n", value);
        return value;
    }
}

void display(struct Queue *q) {
    if (isEmpty(q)) {
        printf("Queue is empty\n");
    } else {
        printf("Queue content: ");
        for (int i = q->front; i <= q->rear; i++) {
            printf("%s ", q->items[i]);
        }
        printf("\n");
    }
}

int main() {
    struct Queue myQueue;
    initializeQueue(&myQueue);

    enqueue(&myQueue, "Apple");
    enqueue(&myQueue, "Banana");
    enqueue(&myQueue, "Orange");

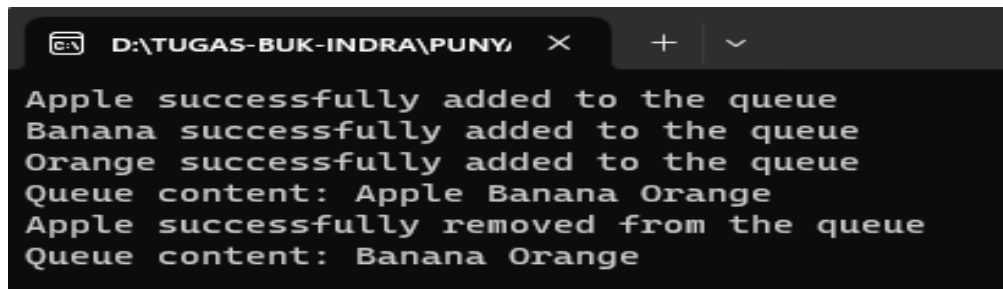
    display(&myQueue);

    dequeue(&myQueue);

    display(&myQueue);

    return 0;
}
```

Output:



```
D:\TUGAS-BUK-INDRA\PUNY\ >
Apple successfully added to the queue
Banana successfully added to the queue
Orange successfully added to the queue
Queue content: Apple Banana Orange
Apple successfully removed from the queue
Queue content: Banana Orange
```

Analisa :

Program ini adalah implementasi sederhana dari struktur data queue dengan elemen bertipe string (array karakter) dalam bahasa C. Berikut adalah analisis singkatnya:

1. Struktur Data Queue:
 - Program menggunakan struktur data Queue untuk menyimpan elemen-elemen bertipe string.
 - Struktur Queue memiliki array items sebagai wadah elemen-elemen bertipe string, serta variabel front dan rear untuk menandai posisi depan dan belakang dari queue.
2. Fungsi-fungsi Utama:
 - initializeQueue: Menginisialisasi queue dengan mengatur front dan rear ke -1.
 - isFull: Memeriksa apakah queue penuh.
 - isEmpty: Memeriksa apakah queue kosong.
 - enqueue: Menambahkan elemen bertipe string ke dalam queue.
 - dequeue: Menghapus elemen bertipe string dari depan queue dan mengembalikan nilainya.
 - display: Menampilkan elemen-elemen bertipe string dalam queue.
3. Implementasi Queue dengan String:
 - Fungsi enqueue menambahkan elemen bertipe string ke dalam queue.
 - Fungsi dequeue menghapus elemen bertipe string dari depan queue dan mengembalikan nilainya.
 - Karena menggunakan array karakter untuk menyimpan string, fungsi strcpy digunakan untuk menyalin nilai string.
4. Fungsi Utama (main):
 - Membuat queue baru menggunakan initializeQueue.
 - Menambahkan beberapa elemen bertipe string ke dalam queue menggunakan enqueue.
 - Menampilkan elemen bertipe string dalam queue menggunakan display.
 - Menghapus elemen bertipe string dari depan queue menggunakan dequeue.
 - Menampilkan kembali elemen bertipe string dalam queue menggunakan display.

1.3. PENUTUP

A. Kesimpulan

Dalam praktikum ini, kami tidak hanya mempelajari konsep dasar pemrograman C/C++ tetapi juga menerapkannya dalam konteks struktur data dan fungsi. Fokus utama praktikum adalah pada pemahaman dan penerapan konsep antrian (queue) dalam bahasa pemrograman C/C++. Konsep antrian sangat penting, terutama karena mengikuti prinsip FIFO, yang dapat digunakan untuk mengatasi berbagai permasalahan terkait pengelolaan data. Kami telah mendalami operasi dasar seperti enqueue (penyisipan) dan dequeue (penghapusan), serta fungsi tambahan seperti front (elemen di depan) dan rear (elemen di belakang) pada struktur data queue. Contoh kode yang telah disajikan memberikan gambaran praktis tentang bagaimana menggunakan program untuk mengimplementasikan dan memanipulasi queue. Dengan demikian, kami dapat lebih memahami bagaimana antrian dapat diintegrasikan ke dalam solusi program kami. Praktikum ini memberikan kesempatan bagi kami untuk mengasah keterampilan pemrograman kami dan meningkatkan kemampuan dalam memahami dan mengelola struktur data secara efisien. Penerapan konsep antrian juga membantu kami memahami bagaimana mengatasi situasi di mana data harus diproses sesuai dengan urutan kedatangan mereka. Secara keseluruhan, praktikum ini tidak hanya menjadi landasan bagi pemahaman dasar dalam pemrograman C/C++, tetapi juga membuka wawasan tentang penggunaan struktur data seperti antrian dalam menyelesaikan masalah pemrograman yang lebih kompleks.

DAFTAR PUSTAKA

<https://www.ismynr.xyz/2019/02/penjelasan-contoh-program-queue-cpp.html?m=1>

<https://www.mukhlis.com/2019/10/contoh-program-queue-antrian-bank-dalam.html?m=1%5B3>

<https://sites.google.com/a/mhs.uinjkt.ac.id/romadhon-byar/artikel/Contoh-Program-Queue-dengan-bahasa-C>