

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

MODUL 6

ADT Queue Representasi Kontigu “Non Circular”

DISUSUN OLEH :

AJI KARTIKO HARTANTO - 2350081062



**PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS DAN INFORMATIKA
UNIVERSITAS JENDERAL ACHMAD YANI
TAHUN 2024**

DAFTAR ISI

DAFTAR GAMBAR	ii
BAB I. HASIL PRAKTIKUM.....	1
I.1 Latihan.....	1
I.1.A. Source Code Boolean.h.....	1
I.1.B. Source Code Queue.h.....	1
I.1.C. Source Code Queue.c.....	3
I.1.D. Source Code mqueue.c.....	7
I.1.E. Hasil	10
I.1.F. Analisa	10
BAB II. TUGAS PRAKTIKUM	11
II.1 Tugas	11
II.1.A. Source Code Boolean.h	11
II.1.B. Source Code queue2.h.....	11
II.1.C. Source Code queue2.c	13
II.1.D. Source Code mqueue2.c	17
II.1.E. Hasil	20
II.1.F. Analisa	20
BAB III. KESIMPULAN	21

DAFTAR GAMBAR

Gambar I.1 Output Program mqueue.c	10
Gambar II.1 Output Program Tugas mqueue alternative 2.....	20

BAB I. HASIL PRAKTIKUM

I.1 Latihan

I.1.A. Source Code Boolean.h

```
/*
    Program      : boolean.h
    Deskripsi    : Header file dari boolean
*/

#ifndef boolean_H
#define boolean_H
#define true 1
#define false 0
#define boolean unsigned char

#endif
```

I.1.B. Source Code Queue.h

```
/*
    Program      : queue1.h
    Author       : 2350081062, Aji Kartiko Hartanto
    Kelas        : C
    Deskripsi    : Header file dari queue
    Tanggal      : 22 Mei 2024
*/

#ifndef QUEUE_H
#define QUEUE_H

#include "boolean.h"

#include <stdio.h>
#include <conio.h>

#define Nil 0
```

```

#define MaxEl 10
#define IdxUndef (-999)
#define Head(Q) (Q).HEAD
#define Tail(Q) (Q).TAIL
#define InfoHead(Q) (Q).T[(Q).HEAD]
#define InfoTail(Q) (Q).T[(Q).TAIL]
#define InfoElm(Q) (Q).T[i]
#define InfoNextElm(Q) (Q).T[i + 1]

// Definisi Queue
typedef int infotype;
typedef int address;

typedef struct {
    infotype T[MaxEl + 1];
    address HEAD;
    address TAIL;
} Queue;

// Prototype Queue
// Konstruktor membentuk Queue
void CreateQueue(Queue * Q);

// {Operasi terhadap komponen: selektor Get dan Set} : tidak perlu
sudah di define diatas
int NBElt(Queue Q);

// Destruktor/Dealokator: tidak perlu

// {Kelompok operasi pada Queue}
boolean IsQueueEmpty(Queue Q);

boolean IsQueueFull(Queue Q);

// Menambahkan sebuah element ke queue
void AddQueue(Queue *Q, infotype X);

```

```

// Menghapus sebuah element queue
void DelQueue(Queue *Q, infotype *X);

// {Kelompok interaksi dengan i/o device, Baca/Tulis}
void PrintQueueInfo(Queue S);

// Kelompok operasi lain terhadap type
boolean isInfoKetemu(Queue S, infotype X);

address CariElemenQueue(Queue Q, int X);

#endif

```

I.1.C. Source Code Queue.c

```

/*
    Program      : queue1.c
    Author       : 2350081062, Aji Kartiko Hartanto
    Kelas        : C
    Deskripsi    : file .c dari header queue
    Tanggal      : 22 Mei 2024
*/

#include "queue.h"

// Prototype Queue
// Konstruktor membentuk Queue
void CreateQueue(Queue *Q) {
    Head(*Q) = Nil;
    Tail(*Q) = Nil;
}

// {Operasi terhadap komponen: selektor Get dan Set} : tidak perlu
sudah di define diatas
int NBEIlt(Queue Q) {
    if (Head(Q) == 0 && Tail(Q) == 0) {
        return 0;
    } else {

```

```

        return Tail(Q);
    }
}

// Destruktor/Dealokator: tidak perlu

// {Kelompok operasi pada Queue}
boolean IsQueueEmpty(Queue Q) {
    if (Head(Q) == Nil && Tail(Q) == Nil) {
        return true;
    } else {
        return false;
    }
}

boolean IsQueueFull(Queue Q) {
    if (Tail(Q) == MaxEl) {
        return true;
    } else {
        return false;
    }
}

// Menambahkan sebuah element ke queue
void AddQueue(Queue *Q, infotype X) {
    if (IsQueueEmpty(*Q)) {
        Head(*Q)++;
        InfoHead(*Q) = X;

        Tail(*Q) = Head(*Q);
    } else {
        if (!IsQueueFull(*Q)) {
            Tail(*Q)++;
            InfoTail(*Q) = X;
        } else {
            printf("Stack penuh");
        }
    }
}

```



```

    }
}

// Menghapus sebuah element queue
void DelQueue(Queue *Q, infotype *X) {
    // kamus
    int i;

    // algoritma
    if (!IsEmpty(*Q)) {
        *X = InfoHead(*Q);

        for (i = Head(*Q); i <= Tail(*Q); i++) {
            InfoElm(*Q) = InfoNextElm(*Q);
        }

        Tail(*Q)--;
    } else {
        printf("Queue kosong");
    }
}

// {Kelompok interaksi dengan i/o device, Baca/Tulis}
void PrintQueueInfo(Queue S) {
    // kamus
    int i;

    // algoritma
    if (!IsEmpty(S)) {
        for (i = Head(S); i <= Tail(S); i++) {
            printf("[%d] <- ", InfoElm(S));
        }

        printf("Null");
    }
}

```

```

// Kelompok operasi lain terhadap type
boolean isInfoKetemu(Queue S, infotype X) {
    // kamus
    int i;

    // algoritma
    if (IsEmpty(S)) {
        return false;
    } else {
        for (i = 1; i <= Tail(S); i++) {
            if (InfoElm(S) == X) {
                return true;
            }
        }

        return false;
    }
}

address CariElemenQueue(Queue Q, int X) {
    // kamus
    int i;

    // algoritma
    if (IsEmpty(Q)) {
        // jika tabel kosong return Index undefined
        return IdxUndef;
    } else {
        // jika ditemukan return address yaitu i
        for (i = 1; i <= Tail(Q); i++) {
            if (InfoElm(Q) == X) {
                return i;
            }
        }

        // jika data x tidak ditemukan return Index undefined
        return IdxUndef;
    }
}

```

```
}  
  
}
```

I.1.D. Source Code mqueue.c

```
/*  
    Program      : mqueue.c  
    Author       : 2350081062, Aji Kartiko Hartanto  
    Kelas        : C  
    Deskripsi     : Main driver dari prototype queue  
    Tanggal      : 22 Mei 2024  
*/  
  
#include "queue.c"  
  
int main() {  
    // Kamus  
    Queue queue;  
    infotype elemen, info;  
  
    // Algoritma  
    CreateQueue(&queue);  
  
    // cek apakah queue kosong  
    printf("Apakah queue kosong?\n");  
    if (IsEmptyQueue(queue)) {  
        printf("Queue masih kosong");  
    } else {  
        printf("Queue tidak kosong");  
    }  
  
    // Menambahkan queue  
    printf("\n\n");  
    printf("Menambah queue");  
    AddQueue(&queue, 1);  
    AddQueue(&queue, 2);  
    AddQueue(&queue, 3);  
    AddQueue(&queue, 4);  
}
```

```

AddQueue(&queue, 5);

// cetak queue
printf("\n");
PrintQueueInfo(queue);

// cek apakah queue penuh
printf("\n\n");
if (IsQueueFull(queue)) {
    printf("Queue penuh");
} else {
    printf("Queue belum penuh");
}

// Menambahkan queue
printf("\n\n");
printf("Menambah queue");
AddQueue(&queue, 6);
AddQueue(&queue, 7);

// cetak queue
printf("\n");
PrintQueueInfo(queue);

// Menghapus queue
printf("\n\n");
printf("Menghapus queue");
DelQueue(&queue, &elemen);

// cetak queue
printf("\n");
PrintQueueInfo(queue);

// Menghapus queue
printf("\n\n");
printf("Menghapus queue");
DelQueue(&queue, &elemen);

```

```

    // cetak queue
    printf("\n");
    PrintQueueInfo(queue);

    // cek apakah x berada pada queue
    printf("\n\n");
    if (isInfoKetemu(queue, 4)) {
        printf("Elemen yang anda masukan terdapat pada queue");
    } else {
        printf("Elemen yang anda masukan tidak terdapat pada
queue");
    }

    // mencari elemen x yang terdapat pada queue
    info = CariElemenQueue(queue, 4);

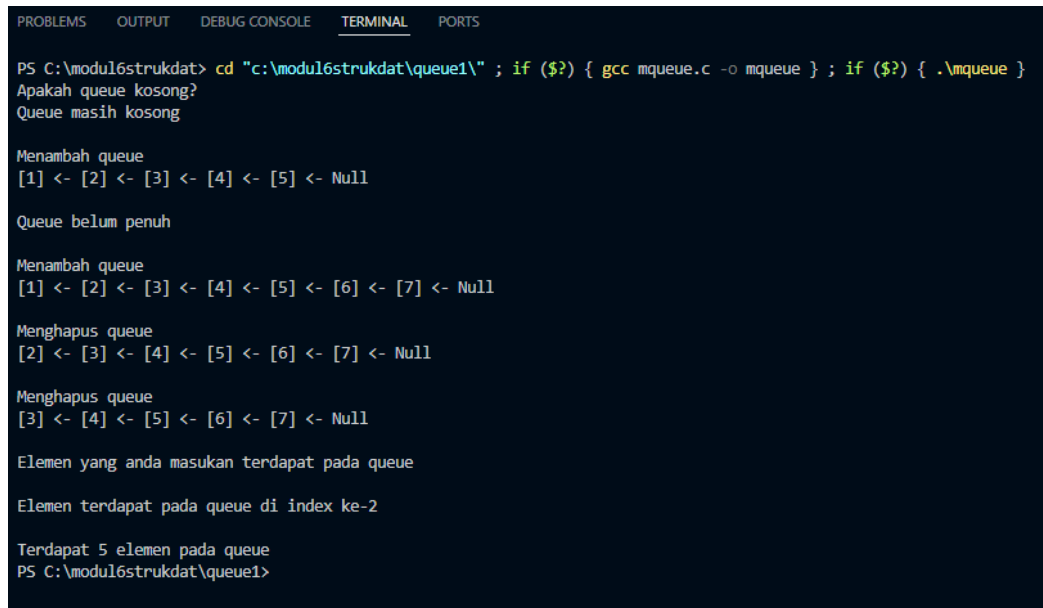
    printf("\n\n");
    if (info != IdxUndef) {
        printf("Elemen terdapat pada queue di index ke-%d", info);
    } else {
        printf("Elemen tidak terdapat pada queue, index %d", info);
    }

    // mengirim banyak elemen queue
    printf("\n\n");
    printf("Terdapat %d elemen pada queue", NBElmt(queue));

    return 0;
}

```

I.1.E. Hasil



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\modul6strukdat> cd "c:\modul6strukdat\queue1\" ; if ($?) { gcc mqueue.c -o mqueue } ; if ($?) { .\mqueue }
Apakah queue kosong?
Queue masih kosong

Menambah queue
[1] <- [2] <- [3] <- [4] <- [5] <- Null

Queue belum penuh

Menambah queue
[1] <- [2] <- [3] <- [4] <- [5] <- [6] <- [7] <- Null

Menghapus queue
[2] <- [3] <- [4] <- [5] <- [6] <- [7] <- Null

Menghapus queue
[3] <- [4] <- [5] <- [6] <- [7] <- Null

Elemen yang anda masukan terdapat pada queue

Elemen terdapat pada queue di index ke-2

Terdapat 5 elemen pada queue
PS C:\modul6strukdat\queue1>
```

Gambar I.1 Output Program mqueue.c

I.1.F. Analisa

Pada program latihan adt queue ini, kita menggunakan metode yang mana Ketika kita menghapus elemen dari head, head bergeser maju, dan ketika elemen baru ditambahkan, tail bergeser maju. Metode ini memanfaatkan ruang memori secara efisien, namun memerlukan penanganan khusus ketika tail mencapai akhir array dan head berada di awal array agar tidak terjadi kesalahan dalam pengelolaan data di dalam queue.

BAB II. TUGAS PRAKTIKUM

II.1 Tugas

II.1.A. Source Code Boolean.h

```
/* Program      : boolean.h
   Deskripsi     : Header file boolean
*/

#ifndef boolean_H
#define boolean_H
#define true 1
#define false 0
#define boolean unsigned char
#endif
```

II.1.B. Source Code queue2.h

```
/*
   Program      : queue2.h
   Author       : 2350081062, Aji Kartiko Hartanto
   Kelas       : C
   Deskripsi    : Header file dari prototype queue
   Tanggal     : 22 Mei 2024
*/

#ifndef QUEUE_H
#define QUEUE_H

#include "boolean.h"

#include <stdio.h>
#include <conio.h>

#define Nil 0
#define MaxEl 10
#define IdxUndef (-999)
```

```

#define Head(Q) (Q).HEAD
#define Tail(Q) (Q).TAIL
#define InfoHead(Q) (Q).T[(Q).HEAD]
#define InfoTail(Q) (Q).T[(Q).TAIL]
#define InfoElm(Q) (Q).T[i]

// Definisi Queue
typedef int infotype;
typedef int address;

typedef struct {
    infotype T[MaxEl + 1];
    address HEAD;
    address TAIL;
} Queue;

// Prototype Queue
// Konstruktor membentuk Queue
void CreateQueue(Queue * Q);

// {Operasi terhadap komponen: selektor Get dan Set} : tidak perlu
sudah di define diatas
int NBElt(Queue Q);

// Destruktor/Dealokator: tidak perlu

// {Kelompok operasi pada Queue}
boolean IsQueueEmpty(Queue Q);

boolean IsQueueFull(Queue Q);

// Menambahkan sebuah element ke queue
void AddQueue(Queue *Q, infotype X);

// Menghapus sebuah element queue
void DelQueue(Queue *Q, infotype *X);

```



```
// {Kelompok interaksi dengan i/o device, Baca/Tulis}
void PrintQueueInfo(Queue S);

// Kelompok operasi lain terhadap type
boolean isInfoKetemu(Queue S, infotype X);

address CariElemenQueue(Queue Q, int X);

#endif
```

II.1.C. Source Code queue2.c

```
/*
    Program      : queue2.c
    Author       : 2350081062, Aji Kartiko Hartanto
    Kelas        : C
    Deskripsi     : file .c dari file header queue
    Tanggal      : 23 Mei 2024
*/

#include "queue.h"

// Prototype Queue
// Konstruktor membentuk Queue
void CreateQueue(Queue *Q) {
    Head(*Q) = Nil;
    Tail(*Q) = Nil;
}

// {Operasi terhadap komponen: selektor Get dan Set} : tidak perlu
sudah di define diatas
int NBElmt(Queue Q) {
    if (Head(Q) == 0 && Tail(Q) == 0) {
        return 0;
    } else {
        return Tail(Q) + 1 - Head(Q);
    }
}
}
```

```

// Destruktor/Dealokator: tidak perlu

// {Kelompok operasi pada Queue}
boolean IsQueueEmpty(Queue Q) {
    if (Head(Q) == Nil && Tail(Q) == Nil) {
        return true;
    } else {
        return false;
    }
}

boolean IsQueueFull(Queue Q) {
    if (Tail(Q) == MaxEl) {
        return true;
    } else {
        return false;
    }
}

// Menambahkan sebuah element ke queue
void AddQueue(Queue *Q, infotype X) {
    if (IsQueueEmpty(*Q)) {
        Head(*Q)++;
        InfoHead(*Q) = X;

        Tail(*Q) = Head(*Q);
    } else {
        if (!IsQueueFull(*Q)) {
            Tail(*Q)++;
            InfoTail(*Q) = X;
        } else {
            printf("Stack penuh");
        }
    }
}

```

```

// Menghapus sebuah element queue
void DelQueue(Queue *Q, infotype *X) {
    // kamus

    // algoritma
    if (!IsEmpty(*Q)) {
        if (Head(*Q) == Tail(*Q)) {
            *X = InfoHead(*Q);

            CreateQueue(Q);
        } else {
            *X = InfoHead(*Q);
            Head(*Q)++;
        }
    } else {
        printf("Queue kosong");
    }
}

// {Kelompok interaksi dengan i/o device, Baca/Tulis}
void PrintQueueInfo(Queue S) {
    // kamus
    int i;

    // algoritma
    if (!IsEmpty(S)) {
        for (i = Head(S); i <= Tail(S); i++) {
            printf("[%d] <- ", InfoElm(S));
        }

        printf("Null");
    }
}

// Kelompok operasi lain terhadap type
boolean isInfoKetemu(Queue S, infotype X) {
    // kamus

```

```

int i;

// algoritma
if (IsQueueEmpty(S)) {
    return false;
} else {
    for (i = 1; i <= Tail(S); i++) {
        if (InfoElm(S) == X) {
            return true;
        }
    }

    return false;
}
}

address CariElemenQueue(Queue Q, int X) {
    // kamus
    int i;

    // algoritma
    if (IsQueueEmpty(Q)) {
        // jika tabel kosong return Index undefine
        return IdxUndef;
    } else {
        // jika ditemukan return addres yaitu i
        for (i = 1; i <= Tail(Q); i++) {
            if (InfoElm(Q) == X) {
                return i;
            }
        }

        // jika data x tidak ditemukan return Index undefine
        return IdxUndef;
    }
}
}

```

II.1.D. Source Code mqueue2.c

```
/*
    Program      : mqueue2.c
    Author       : 2350081062, Aji Kartiko Hartanto
    Kelas        : C
    Deskripsi    : Main driver dari prototype queue
    Tanggal      : 23 Mein 2024
*/

#include "queue.c"

int main() {
    // Kamus
    Queue queue;
    infotype elemen, info;

    // Algoritma
    CreateQueue(&queue);

    // cek apakah queue kosong
    printf("Apakah queue kosong?\n");
    if (IsEmptyQueue(queue)) {
        printf("Queue masih kosong");
    } else {
        printf("Queue tidak kosong");
    }

    // Menambahkan queue
    printf("\n\n");
    printf("Menambah queue");
    AddQueue(&queue, 1);
    AddQueue(&queue, 2);
    AddQueue(&queue, 3);
    AddQueue(&queue, 4);
    AddQueue(&queue, 5);

    // cetak queue
```

```

printf("\n");
PrintQueueInfo(queue);

// cek apakah queue penuh
printf("\n\n");
if (IsQueueFull(queue)) {
    printf("Queue penuh");
} else {
    printf("Queue belum penuh");
}

// Menambahkan queue
printf("\n\n");
printf("Menambah queue");
AddQueue(&queue, 6);
AddQueue(&queue, 7);

// cetak queue
printf("\n");
PrintQueueInfo(queue);

// Menghapus queue
printf("\n\n");
printf("Menghapus queue");
DelQueue(&queue, &elemen);

// cetak queue
printf("\n");
PrintQueueInfo(queue);

// Menghapus queue
printf("\n\n");
printf("Menghapus queue");
DelQueue(&queue, &elemen);

// cetak queue
printf("\n");

```

```

PrintQueueInfo(queue);

// Menghapus queue
printf("\n\n");
printf("Menghapus queue");
DelQueue(&queue, &elemen);

// cetak queue
printf("\n");
PrintQueueInfo(queue);

// cek apakah x berada pada queue
printf("\n\n");
if (isInfoKetemu(queue, 4)) {
    printf("Elemen yang anda masukan terdapat pada queue");
} else {
    printf("Elemen yang anda masukan tidak terdapat pada
queue");
}

// mencari elemen x yang terdapat pada queue
info = CariElemenQueue(queue, 4);

printf("\n\n");
if (info != IdxUndef) {
    printf("Elemen terdapat pada queue di index ke-%d", info);
} else {
    printf("Elemen tidak terdapat pada queue, index %d", info);
}

// mengirim banyak elemen pada queue
printf("\n\n");
printf("Terdapat %d elemen pada queue", NBElmt(queue));

return 0;
}

```

II.1.E. Hasil

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\modul6strukdat\queue2> cd "c:\modul6strukdat\queue2\" ; if ($?) { gcc mqueue.c -o mqueue } ; if ($?) { .\mqueue }
Apakah queue kosong?
Queue masih kosong

Menambah queue
[1] <- [2] <- [3] <- [4] <- [5] <- Null

Queue belum penuh

Menambah queue
[1] <- [2] <- [3] <- [4] <- [5] <- [6] <- [7] <- Null

Menghapus queue
[2] <- [3] <- [4] <- [5] <- [6] <- [7] <- Null

Menghapus queue
[3] <- [4] <- [5] <- [6] <- [7] <- Null

Menghapus queue
[4] <- [5] <- [6] <- [7] <- Null

Elemen yang anda masukan terdapat pada queue

Elemen terdapat pada queue di index ke-4

Terdapat 4 elemen pada queue
PS C:\modul6strukdat\queue2>
```

Gambar II.1 Output Program Tugas mqueue alternative 2

II.1.F. Analisa

Pada tugas program queue yang menggunakan metode kedua ini sebenarnya mirip dengan program latihan yang pertama, yang membedakannya adalah pada cara alternative yang kedua yaitu dengan metode penghapusan nya yaitu head yang bergerak mendekati tail.

BAB III. KESIMPULAN

. pada modul praktikum mempelajari ADT Queue dengan representasi kontigu non-circular dalam modul praktikum ini. Perwakilan ini menggunakan array linear dengan dua pointer, head dan tail. Head menunjuk elemen terdepan untuk dequeue, sementara tail menunjuk posisi untuk enqueue elemen baru. Meskipun masih ada ruang di awal array, proses enqueue tidak dapat dilakukan ketika tail array mencapai akhir.

Dengan implementasi yang sederhana, representasi kontigu non-circular kurang efisien dalam penggunaan ruang memori karena ruang memori terbuang setelah tail mencapai akhir array. Oleh karena itu, peserta akan mempelajari cara mengatasi masalah penggunaan ruang memori yang kurang efisien ini, seperti mengalokasi ulang memori ketika queue mendekati kapasitas maksimumnya. Pemahaman ini sangat penting untuk meningkatkan efisiensi dan kinerja implementasi struktur queue data.