

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

MODUL 7

ADT Queue Representasi Kontigu “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 Queueecircular.h	1
I.1.C. Source Code Queueecircular.c.....	4
I.1.D. Source Code mqueueecircular.c	8
I.1.E. Hasil	11
I.1.F. Analisa	12
BAB II. TUGAS PRAKTIKUM	13
II.1 Tugas (berkelompok)	13
II.1.A. Source Code Boolean.h	13
II.1.B. Source Code queueecircular2.h.....	13
II.1.C. Source Code queueecircular2.c.....	16
II.1.D. Source Code mqueueecircular2.c.....	20
II.1.E. Hasil	22
II.1.F. Analisa	22
BAB III. KESIMPULAN	23

DAFTAR GAMBAR

Gambar I.1 Output Program mqueue circular.c	11
Gambar II.1 Output Program Tugas mqueue circular 2	22

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 Queuecircular.h

```
/*
    Program      : queuecircular.h
    Author       : 2350081062, Aji Kartiko Hartanto
    Kelas       : C
    Deskripsi    : Header file dari prototype queue
    Tanggal     : 29 Mei 2024
*/

#ifndef _QUEUECIRCULAR_H
#define _QUEUECIRCULAR_H
#include "boolean.h"
#include <stdio.h>
#include <conio.h>

#define Nil 0
#define MaxEl 10
/*    nil adalah queue dengan elemen kosong
```

```

    karena indeks dalam bhs C dimulai dari 0 maka tabel dg indeks
0 tidak dipakai */

/* Jika Q adalah Queue maka akses elemen : */
/* Q.T[(Q.HEAD)] untuk mengakses elemen pertama */
/* Q.T[(Q.TAIL)] untuk mengakses elemen terakhir */
/* Q.HEAD adalah alamat elemen pertama */
/* Q.TAIL adalah alamat elemen terakhir */
/* Definisi akses dengan Selektor : Set dan Get */

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

/** Definisi ABSTRACT DATA TYPE Tab Integer **/
/* Definisi Queue :
/* Queue kosong: Head=Nil, TAIL=Nil
Atau Head = Tail + 1 untuk alternatif 2
Elemen yang dipakai menyimpan nilai Stack T[1]..T[MaxEl]
Jika Q adalah Queue maka akses elemen
Q.T[(Q.HEAD)] untuk mengakses elemen pertama
Q.T[(Q.TAIL)] untuk mengakses elemen terakhir
Q.HEAD adalah alamat elemen pertama
Q.TAIL adalah alamat elemen terakhir
*/

typedef int infotype; /*elemen queue bertipe integer*/
typedef int address; /* indeks tabel */

/* Versi pembentukan Queue -> dengan menyimpan tabel dan alamat head
maupun
tail secara eksplisit */

typedef struct { infotype T[MaxEl+1]; /* tabel penyimpan elemen */
    address HEAD; /* alamat HEAD: elemen pertama */
    address TAIL; /* alamat TAIL: elemen terakhir */

```

```

} Queue;

/* Prototype Queue */
/* Konstruktor membentuk Queue */
void CreateQueue(Queue *Q);
/*    I.S : Q terdefinisi tidak diketahui isinya
      F.S : Q diinisialisasi dengan HEAD(Q)=nil, TAIL(Q)=nil */

/** { Operasi terhadap komponen : selektor Get dan Set } : tidak
perlu sudah di define diatas**/
int NBElmt(Queue Q);
/* Mengirimkan banyaknya elemen queue. Mengirimkan 0 jika Q kosong
*/

/* Destruktor/Dealokator: tidak perlu */

/** { KELOMPOK OPERASI pada Queue} **/
/***** Predikat Untuk test keadaan KOLEKSI **/
boolean IsQueueEmpty (Queue Q);
/* Mengirim true jika Queue kosong: lihat definisi di atas */

boolean IsQueueFull(Queue Q);
/*    Mengirim true jika tabel penampung nilai elemen stack penuh
yaitu jumlah
      elemen sudah mencapai MaxEl : lihat definisi diatas */

int NBElmt(Queue Q);
/* Mengirimkan banyaknya elemen queue. Mengirimkan 0 jika Q kosong
*/

/***** Menambahkan sebuah elemen ke Queue *****/
void AddQueue (Queue *Q, infotype X);
/* Proses: Menambahkan X pada Q dengan aturan FIFO */
/* I.S : Q terdefinisi, mungkin kosong, dan Q penampung elemen queue
tidak penuh.
      F.S : F.S. X menjadi TAIL yang baru, TAIL "maju"
      Jika Tail(Q)=MaxEl+1 maka Tail(Q) diset =1

```

```

*/

/***** Menghapus sebuah elemen Queue *****/
void DelQueue (Queue *Q, infotype *X);
/* Proses: Menghapus X pada Q dengan aturan FIFO */
/* I.S : Q terdefinisi, dan Q tidak kosong
   F.S : Q berkurang satu elemen didepan disimpan pada X
   Head(Q) "maju" selama Head(Q) < MaxEl,
   Tetapi Jika Head(Q)=MaxEl+1 maka Tail(Q) diset =1
*/

/** { KELOMPOK Interaksi dengan I/O device, BACA/TULIS } **/
void PrintQueueInfo (Queue S);
/*   I.S : S terdefinisi, mungkin kosong
      F.S : Jika Queue tidak kosong, semua info yang disimpan pada
elemen queue diprint.
*/

/** KELOMPOK OPERASI LAIN TERHADAP TYPE **/
boolean isInfoKetemu(Queue S, infotype x);
/* mengirim true jika x ada pada Q */

address CariElemenQueue(Queue Q, int X);
// Search apakah ada elemen tabel T yang bernilai X)
// Jika x ada pada Q, menghasilkan address terkecil
// Jika tidak ada, mengirimkan IdXUndef)
// Menghasilkan indeks tak terdefinisi(IdXUndef) jika Queue kosong)
// Memakai Skema search DENGAN boolean Found}

#endif

```

I.1.C. Source Code Queuecircular.c

```

/*
    Program      : queuecircular.c
    Author       : 2350081062, Aji Kartiko Hartanto
    Kelas        : C
    Deskripsi    : Body file dari prototype queue

```


Tanggal : 29 Mei 2024

```
*/

#include "queuecircular.h"

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

/* Mengirimkan banyaknya elemen queue. Mengirimkan 0 jika Q kosong */
int NBElt(Queue Q) {
    if (IsEmpty(Q)) {
        return 0;
    } else if (Head(Q) <= Tail(Q)) {
        return Tail(Q) - Head(Q) + 1;
    } else {
        return (MaxEl - Head(Q) + 1) + Tail(Q);
    }
}

/* Mengirim true jika Queue kosong: lihat definisi di atas */
boolean IsEmpty(Queue Q) {
    return (Head(Q) == Nil && Tail(Q) == Nil);
}

/* Mengirim true jika tabel penampung nilai elemen stack penuh yaitu
jumlah elemen sudah mencapai MaxEl */
boolean IsFull(Queue Q) {
    return (NBElt(Q) == MaxEl);
}

/* Proses: Menambahkan X pada Q dengan aturan FIFO */
void AddQueue(Queue *Q, infotype X) {
    if (IsEmpty(*Q)) {
```

```

        Head(*Q) = 1;
        Tail(*Q) = 1;
        InfoTail(*Q) = X;
    } else {
        if (Tail(*Q) == MaxEl) {
            Tail(*Q) = 1;
        } else {
            Tail(*Q)++;
        }
        InfoTail(*Q) = X;
    }
}

/* Proses: Menghapus X pada Q dengan aturan FIFO */
void DelQueue(Queue *Q, infotype *X) {
    *X = InfoHead(*Q);
    if (Head(*Q) == Tail(*Q)) {
        Head(*Q) = Nil;
        Tail(*Q) = Nil;
    } else {
        if (Head(*Q) == MaxEl) {
            Head(*Q) = 1;
        } else {
            Head(*Q)++;
        }
    }
}

/* I.S : S terdefinisi, mungkin kosong
   F.S : Jika Queue tidak kosong, semua info yang disimpan pada
   elemen queue diprint. */
void PrintQueueInfo(Queue Q) {
    if (IsQueueEmpty(Q)) {
        printf("Queue is empty\n");
    } else {
        address i = Head(Q);
        while (1) {

```

```

        printf("|%d|", Q.T[i]);
        if (i == Tail(Q)) break;
        if (i == MaxEl) {
            i = 1;
        } else {
            i++;
        }
    }
    printf("\n");
}

/* mengirim true jika x ada pada Q */
boolean isInfoKetemu(Queue Q, infotype X) {
    if (IsEmpty(Q)) {
        return false;
    } else {
        address i = Head(Q);
        do {
            if (Q.T[i] == X) {
                return true;
            }
            if (i == MaxEl) {
                i = 1;
            } else {
                i++;
            }
        } while (i != Tail(Q));
        return (Q.T[Tail(Q)] == X);
    }
}

/* Search apakah ada elemen tabel T yang bernilai X
   Jika x ada pada Q, menghasilkan address terkecil
   Jika tidak ada, mengirimkan Nil
   Memakai Skema search DENGAN boolean Found */
address CariElemenQueue(Queue Q, int X) {

```

```

    if (IsQueueEmpty(Q)) {
        return Nil;
    } else {
        address i = Head(Q);
        do {
            if (Q.T[i] == X) {
                return i;
            }
            if (i == MaxEl) {
                i = 1;
            } else {
                i++;
            }
        } while (i != Tail(Q));
        if (Q.T[Tail(Q)] == X) {
            return Tail(Q);
        } else {
            return Nil;
        }
    }
}

```

I.1.D. Source Code mqueuecircular.c

```

/*
    Program      : mqueuecircular.c
    Author       : 2350081062, Aji Kartiko Hartanto
    Kelas        : C
    Deskripsi    : Main Driver dari prototype queue
    Tanggal      : 29 Mei 2024
*/
#include "queuecircular.h"
#include <stdio.h>

void PrintMenu() {
    printf("==== { MENU PILIHAN } =====\n");
    printf("1. ADD QUEUE\n");
    printf("2. DELETE QUEUE\n");
}

```

```

    printf("3. CARI ELEMEN\n");
    printf("0. QUEUE BERHENTI\n");
}
void PrintQueueState(Queue Q) {
    int i;
    printf("\nELEMEN : ");
    PrintQueueInfo(Q);
    if (!IsQueueEmpty(Q)) {
        printf("Nilai HEAD : %d\n", InfoHead(Q));
        printf("HEAD berada pada tabel ke %d\n", Head(Q));
        printf("Nilai TAIL : %d\n", InfoTail(Q));
        printf("TAIL berada pada tabel ke %d\n", Tail(Q));
        printf("Banyak antrian : %d\n", NBElmt(Q));

        // Menghitung rata-rata waktu menunggu (jumlah seluruh
        elemen dibagi dengan jumlah elemen)
        int sum = 0;
        for (i = Head(Q); i != Tail(Q); i = (i % MaxEl) + 1) {
            sum += Q.T[i];
        }
        sum += Q.T[Tail(Q)];
        printf("Waktu rata rata mengantri : %.2f\n", (float)sum /
NBElmt(Q));
    }
}
int main() {
    Queue Q;
    CreateQueue(&Q);
    int pilihan;
    infotype elem;

    do {
        PrintMenu();
        printf("Silahkan masukkan pilihan = ");
        scanf("%d", &pilihan);
        switch (pilihan) {
            case 1:

```

```

        printf("\nMasukkan elemen : ");
        scanf("%d", &elem);
        if (!IsQueueFull(Q)) {
            AddQueue(&Q, elem);
        } else {
            printf("Queue penuh, tidak dapat menambahkan
elemen\n");
        }
        PrintQueueState(Q);
        break;
case 2:
    if (!IsQueueEmpty(Q)) {
        DelQueue(&Q, &elem);
        printf("\nElemen yang dihapus: %d\n", elem);
    } else {
        printf("Queue kosong, tidak ada elemen yang
dapat dihapus\n");
    }
    PrintQueueState(Q);
    break;
case 3:
    printf("\nMasukkan elemen yang dicari : ");
    scanf("%d", &elem);
    address idx = CariElemenQueue(Q, elem);
    if (idx != Nil) {
        printf("Elemen %d ada\n", elem);
        printf("Ditemukan pada indeks ke %d\n", idx);
    } else {
        printf("Elemen %d tidak ditemukan\n", elem);
    }
    break;
case 0:
    printf("Queue berhenti\n");
    break;
default:
    printf("Pilihan tidak valid\n");
    break;

```

```

    }
} while (pilihan != 0);
return 0;
}

```

I.1.E. Hasil

```

C:\strukdat7\queC1\ADT_Queue >
===== { MENU PILIHAN } =====
1. ADD QUEUE
2. DELETE QUEUE
3. CARI ELEMEN
0. QUEUE BERHENTI
Silahkan masukkan pilihan = 1

Masukkan elemen : 1

ELEMEN : |1|
Nilai HEAD : 1
HEAD berada pada tabel ke 1
Nilai TAIL : 1
TAIL berada pada tabel ke 1
Banyak antrian : 1
Waktu rata rata mengantri : 1.00
===== { MENU PILIHAN } =====
1. ADD QUEUE
2. DELETE QUEUE
3. CARI ELEMEN
0. QUEUE BERHENTI
Silahkan masukkan pilihan = 1

Masukkan elemen : 2

ELEMEN : |1||2|
Nilai HEAD : 1
HEAD berada pada tabel ke 1
Nilai TAIL : 2
TAIL berada pada tabel ke 2
Banyak antrian : 2
Waktu rata rata mengantri : 1.50
===== { MENU PILIHAN } =====
1. ADD QUEUE
2. DELETE QUEUE
3. CARI ELEMEN
0. QUEUE BERHENTI
Silahkan masukkan pilihan = 1

Masukkan elemen : 3

ELEMEN : |1||2||3|
Nilai HEAD : 1
HEAD berada pada tabel ke 1
Nilai TAIL : 3
TAIL berada pada tabel ke 3
Banyak antrian : 3
Waktu rata rata mengantri : 2.00
===== { MENU PILIHAN } =====
1. ADD QUEUE
2. DELETE QUEUE
3. CARI ELEMEN
0. QUEUE BERHENTI
Silahkan masukkan pilihan = 2

Elemen yang dihapus: 1

ELEMEN : |2||3|
Nilai HEAD : 2
HEAD berada pada tabel ke 2
Nilai TAIL : 3
TAIL berada pada tabel ke 3
Banyak antrian : 2
Waktu rata rata mengantri : 2.50
===== { MENU PILIHAN } =====
1. ADD QUEUE
2. DELETE QUEUE
3. CARI ELEMEN
0. QUEUE BERHENTI
Silahkan masukkan pilihan = 3

Masukkan elemen yang dicari : 2
Elemen 2 ada
Ditemukan pada indeks ke 2
===== { MENU PILIHAN } =====
1. ADD QUEUE
2. DELETE QUEUE
3. CARI ELEMEN
0. QUEUE BERHENTI
Silahkan masukkan pilihan = 0
Queue berhenti

-----
Process exited after 26.8 seconds with return value 0

```

Gambar I.1 Output Program mqueue circular.c

I.1.F. Analisa

File header `queuecircular.h` dan implementasinya mendefinisikan dan mengelola struktur queue menggunakan metode buffer lingkaran dengan elemen integer. Queue ini memiliki indeks HEAD dan TAIL untuk melacak posisi elemen pertama dan terakhir. Itu dapat menampung hingga elemen `MaxEl`. Fungsi file ini termasuk `CreateQueue`, mengecek apakah queue kosong atau penuh (`IsQueueEmpty`, `IsQueueFull`), menambah dan menghapus elemen (`AddQueue`, `DelQueue`), mencetak elemen, dan mencari elemen (`isInfoKetemu`, `CariElemenQueue`). Dengan menangani batasan array statis secara dinamis, implementasi ini memastikan operasi FIFO yang efisien. Ini juga menjaga queue tetap optimal dalam penggunaan memori dan akses elemen.

BAB II. TUGAS PRAKTIKUM

II.1 Tugas (berkelompok)

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 queuecircular2.h

```
/*
   Program      : queuecircular2.h
   Author       : 2350081062 Aji Kartiko Hartanto
                : 2350081064 Jarwo Eddy Wicaksono
                : 2350081079 Rifqi Fauzi Anwar

   Kelas       : C
   Deskripsi    : Header File ADT Queue Circular
   Tanggal     : 29 Mei 2024
*/

#ifndef _QUEUE2_H
#define _QUEUE2_H
#include "boolean.h"
#include <stdio.h>
#include <conio.h>

//pendefinisian pointer
#define nil NULL
#define info(P) (P)->info
```

```

#define next(P) (P)->next
#define Head(Q) (Q).HEAD
#define Tail(Q) (Q).TAIL

//Pembentukan tipe Queue
typedef struct tElmQueue *address;
typedef struct tElmQueue{
    int info;
    address next;
} ElmQueue;

typedef struct{
    address HEAD;
    address TAIL;
} Queue;

//Prototype Queue/ Primitif Queue dengan Pointer
//Konstruktor
void CreateQueue(Queue *Q);
/* I.S : Q terdefinisi tidak diketahui isinya
   F.S : Q diinisialisasi dengan HEAD(Q) = Nil, TAIL(Q) = Nil
*/

address Alokasi(int X);
/* Mengirim sebuah address jika alokasi type Queue berhasil */
/* P direlease dari memori */

//Operasi Queue
void AddQueue(Queue *Q, int X);
/* I.S : Q terdefinisi, mungkin kosong, dan
   F.S : Q bertambah sebuah elemen bernilai X dibelakang
   Proses : Head(Q) merupakan elemen terakhir dari Q
*/

void DelQueue(Queue *Q, int *Y);
/* I.S : Q terdefinisi sembarang tidak kosong
   F.S : Q berkurang satu elemen didepan disimpan pada Y

```

```

        Proses : Y = info(HEAD), HEAD(Q) = next(HEAD(Q))
                Dealokasi elemen pertama
    */

//Operasi I/O terhadap Queue
void CetakQueue(Queue Q);
/* I.S : Q terdefinisi sembarang tidak kosong
   F.S : Elemen Queue dicetak dilayar
*/

//Operasi boolean dan relasi terhadap Queue
boolean IsQueueEmpty(Queue Q);
/* Mengirim true jika Queue kosong HEAD(Q) = nil
   false sebaliknya*/

boolean CariElemenQueue(Queue Q, int X);
/* Mengirim true jika elemen X ada pada Q,
   dan false jika X tidak ditemukan pada Q
   Skema pencarian dengan boolean
*/

//Operasi tambahan terhadap Queue
int GetIndex(Queue Q, address P);
/* Menentukan address dari nilai P */

void PrintQueueData(Queue Q);
#endif

```

II.1.C. Source Code queuecircular2.c

```
/*
    Program          : queuecircular2.c
    Author           : 2350081062 Aji Kartiko Hartanto
                    : 2350081064 Jarwo Eddy Wicaksono
                    : 2350081079 Rifqi Fauzi Anwar

    Kelas           : C
    Deskripsi        : Body File ADT Queue Circular
    Tanggal          : 29 Mei 2024
*/

#include "queuecircular2.h"
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

//Konstruktor
void CreateQueue(Queue *Q){
    Head(*Q) = Tail(*Q) = nil;
}

address Alokasi(int X){
    //Kamus Lokal
    ElmQueue *P;
    //Algoritma
    P = (ElmQueue*)malloc(sizeof(ElmQueue));
    if(P != nil){
        info(P) = X;
        next(P) = nil;
        return P;
    }
    else{
        return nil;
    }
}

//Operasi Queue
```

```

void AddQueue(Queue *Q, int X){
//Kamus Lokal
    address P;
    ElmQueue *T;
//Algoritma
    P = Alokasi(X);
    T = Tail(*Q);
    if(IsQueueEmpty(*Q)){
        next(P) = Head(*Q);
        next(P) = Tail(*Q);
        Head(*Q) = P;
        Tail(*Q) = P;
    }
    else{
        while (next(T) != nil){
            T = next(T);
        }
        next(T) = P;
        Tail(*Q) = P;
    }
}

void DelQueue(Queue *Q, int *Y){
    address P = Head(*Q);
    *Y = info(P);

    if (Head(*Q) == Tail(*Q)) {
        Head(*Q) = nil;
        Tail(*Q) = nil;
    } else {
        Head(*Q) = next(Head(*Q));
        next(P) = nil;
    }
}

//Operasi I/O terhadap Queue
void CetakQueue(Queue Q){

```

```

//Kamus Lokal
    ElmQueue *P, *S;
//Algoritma
    if(!IsEmptyQueue(Q)){
        P = Head(Q);
        while(P != nil){
            printf(" [ %d ]", info(P));
            P = next(P);
        }
    }
    else{
        printf("[ Empty ]");
    }
}

//Operasi boolean dan relasi terhadap Queue
boolean IsQueueEmpty(Queue Q){
    if(Head(Q) == nil || Tail(Q) == nil){
        return 1;
    } else {
        return 0;
    }
}

boolean CariElemenQueue(Queue Q, int X){
//Kamus Lokal
    address P;
    boolean ketemu;
//Algoritma
    P = Head(Q);
    ketemu = false;
    if(!IsEmptyQueue(Q)){
        while(P != nil && ketemu == false){
            if(info(P) == X){
                return 1;
                ketemu = true;
            }
            P = next(P);
        }
    }
}

```

```

        return 0;

    }

}

//Operasi tambahan terhadap Queue
int GetIndex(Queue Q, address P) {
    int index = 1;
    address current = Head(Q);
    while (current != nil) {
        if (current == P) {
            return index;
        }
        index++;
        current = next(current);
    }
    return -1;
}

void PrintQueueData(Queue Q) {
    printf("\nNilai HEAD = %d\n", info(Head(Q)));
    printf("HEAD berada pada index ke %d\n", GetIndex(Q, Head(Q)));
    printf("Nilai TAIL = %d\n", info(Tail(Q)));
    printf("TAIL berada pada index ke %d\n", GetIndex(Q, Tail(Q)));

    // Menghitung Banyak Antrian
    int count = 0;
    address P = Head(Q);
    while (P != nil) {
        count++;
        P = next(P);
    }
    printf("Banyak Antrian = %d\n", count);

    // Menghitung Rata-Rata
    int total = 0;
    float avg = 0.0;
    P = Head(Q);
    while (P != nil) {
        total += info(P);
    }
}

```

```

        P = next(P);
    }
    if (count > 0) {
        avg = (float)total / count;
    }
    printf("Waktu rata rata mengantri = %.2f\n", avg);
    printf("=====\n");
}

```

II.1.D. Source Code mqueuecircular2.c

```

/*
    Program          : mqueuecircular2.c
    Author           : 2350081062 Aji Kartiko Hartanto
                    : 2350081064 Jarwo Eddy Wicaksono
                    : 2350081079 Rifqi Fauzi Anwar

    Kelas           : C
    Deskripsi        : Main Driver File ADT Queue Circular
    Tanggal          : 29 Mei 2024
*/

#include "queuecircular2.h"
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int main() {
    Queue MyQueue;
    int N, X, choice, cari;
    boolean found;

    CreateQueue(&MyQueue);
    printf("====Menu Pilihan====\n");
    printf("1. Add Queue \n");
    printf("2. Delete Queue \n");
    printf("3. Cari Elemen \n");
    printf("0. Keluar \n");
    printf("=====\n");
}

```

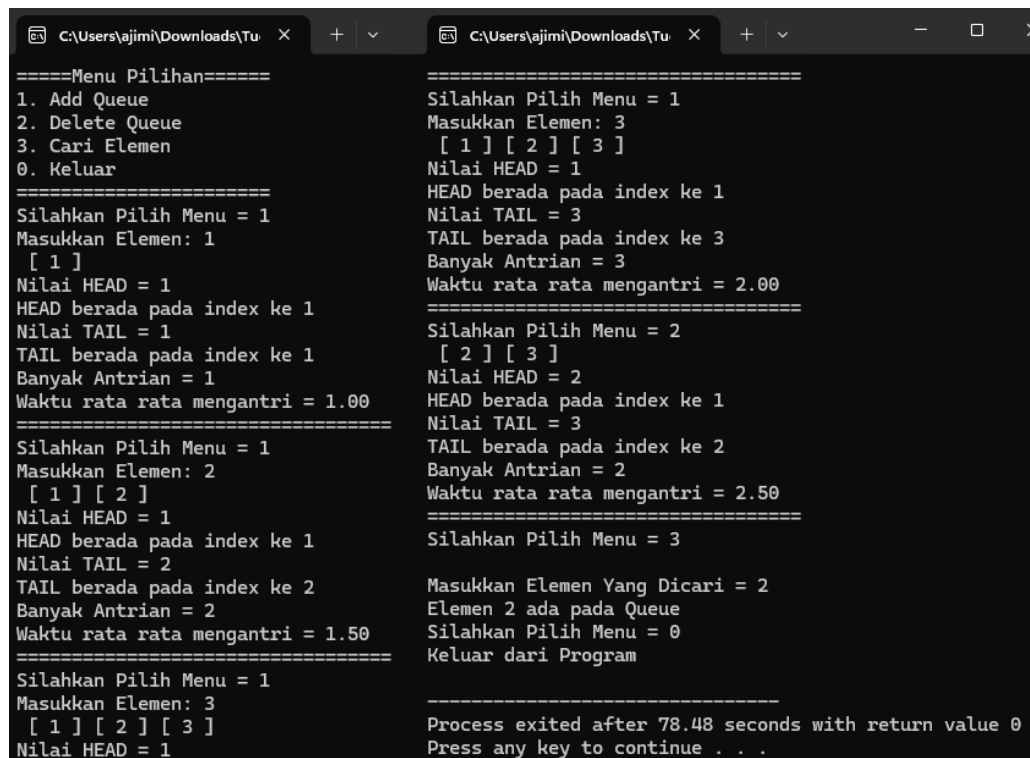


```

do {
    printf("Silahkan Pilih Menu = ");
    scanf("%d", &choice);
    switch (choice) {
        case 1:
            printf("Masukkan Elemen: ");
            scanf("%d", &N);
            AddQueue(&MyQueue, N);
            CetakQueue(MyQueue);
            PrintQueueData(MyQueue);
            break;
        case 2:
            DelQueue(&MyQueue, &X);
            CetakQueue(MyQueue);
            PrintQueueData(MyQueue);
            break;
        case 3:
            printf("\nMasukkan Elemen Yang Dicari = ");
            scanf("%d", &cari);
            if (CariElemenQueue(MyQueue, cari)) {
                printf("Elemen %d ada pada Queue\n", cari);
            } else {
                printf("Elemen %d tidak ada pada Queue\n",
cari);
            }
            break;
        case 0:
            printf("Keluar dari Program \n");
            break;
        default:
            printf("Pilihan tidak ada di Menu \n\n");
            break;
    }
} while (choice != 0);
return 0;
}

```

II.1.E. Hasil



```
====Menu Pilihan====
1. Add Queue
2. Delete Queue
3. Cari Elemen
0. Keluar
=====
Silahkan Pilih Menu = 1
Masukkan Elemen: 1
[ 1 ]
Nilai HEAD = 1
HEAD berada pada index ke 1
Nilai TAIL = 1
TAIL berada pada index ke 1
Banyak Antrian = 1
Waktu rata rata mengantri = 1.00
=====
Silahkan Pilih Menu = 1
Masukkan Elemen: 2
[ 1 ] [ 2 ]
Nilai HEAD = 1
HEAD berada pada index ke 1
Nilai TAIL = 2
TAIL berada pada index ke 2
Banyak Antrian = 2
Waktu rata rata mengantri = 1.50
=====
Silahkan Pilih Menu = 1
Masukkan Elemen: 3
[ 1 ] [ 2 ] [ 3 ]
Nilai HEAD = 1

=====Menu Pilihan=====
Silahkan Pilih Menu = 1
Masukkan Elemen: 3
[ 1 ] [ 2 ] [ 3 ]
Nilai HEAD = 1
HEAD berada pada index ke 1
Nilai TAIL = 3
TAIL berada pada index ke 3
Banyak Antrian = 3
Waktu rata rata mengantri = 2.00
=====
Silahkan Pilih Menu = 2
[ 2 ] [ 3 ]
Nilai HEAD = 2
HEAD berada pada index ke 1
Nilai TAIL = 3
TAIL berada pada index ke 2
Banyak Antrian = 2
Waktu rata rata mengantri = 2.50
=====
Silahkan Pilih Menu = 3

Masukkan Elemen Yang Dicari = 2
Elemen 2 ada pada Queue
Silahkan Pilih Menu = 0
Keluar dari Program

=====
Process exited after 78.48 seconds with return value 0
Press any key to continue . . .
```

Gambar II.1 Output Program Tugas mqueue circular 2

II.1.F. Analisa

Program implementasi mendefinisikan fungsi-fungsi ini, memberikan alokasi memori untuk elemen baru, menambah elemen ke Queue dengan AddQueue, dan menghapus elemen dari Queue dengan DelQueue. IsQueueEmpty dan CariElemenQueue memeriksa kondisi dan mencari elemen dalam Queue, dan fungsi tambahan GetIndex menentukan indeks elemen tertentu. PrintQueueData menampilkan informasi lengkap.

BAB III. KESIMPULAN

Kesimpulan yang dapat diberikan pada praktikum kali ini adalah dengan menggunakan daftar yang terhubung dengan pointer di C, kita memiliki definisi dan implementasi fungsi yang lengkap untuk mengelola Queue. Header file mendefinisikan struktur dasar Queue serta berbagai fungsi operasinya, seperti inisialisasi, alokasi memori, penambahan, penghapusan, dan pemeriksaan elemen.