LEMBAR ASISTENSI

PRAKTIKUM STRUKTUR DATA

LABORATORIUM TEKNIK KOMPUTER

JURUSAN TEKNIK ELEKTRO FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

Judul Praktikum	: QUEUE	
Praktikan	: Alvin Reihansyah Makarim	(2115061083)
Asisten	: Dwindy Monica	(2015061022)
	Muhammad Wafa Al Ausath	(2015061057)
Kelas	: PSTI C	

No.	Catatan	Tanggal	Paraf
1.	Asistensi 1 : - Format - Bab 5 - Kesimpulan - Tugas Akhir		

Bandar Lampung, 2022

.....

NPM.

I. JUDUL PERCOBAAN

QUEUE

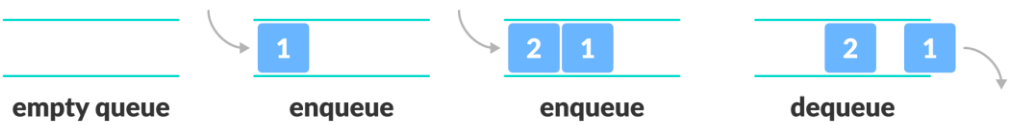
II. TUJUAN PERCOBAAN

Adapun tujuan dari percobaan ini adalah sebagai berikut :

1. Memahami Queue pada Struktur Data
2. Dapat memahami dan menggunakan Queue

III. TEORI DASAR

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). Struktur data queue umumnya digunakan untuk mengelola thread dalam multithreading dan menerapkan sistem antrian prioritas pada program komputer. 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. Seperti stack, queue juga dapat diimplementasikan menggunakan struktur data linked list atau array. Berikut adalah ilustrasi dari queue.



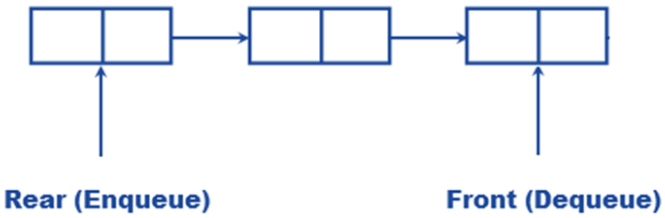
Gambar 1 Ilustrasi queue

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.

Secara umum ada 4 jenis struktur data queue, yaitu:

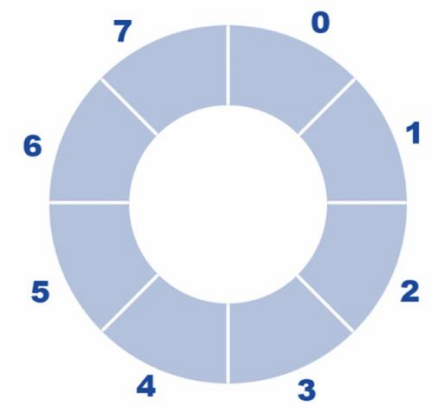
1. Simple queue

Simple queue adalah struktur data queue paling dasar di mana penyisipan item dilakukan di simpul belakang (rear atau tail) dan penghapusan terjadi di simpul depan (front atau head).



Gambar 2 Ilustrasi simple queue

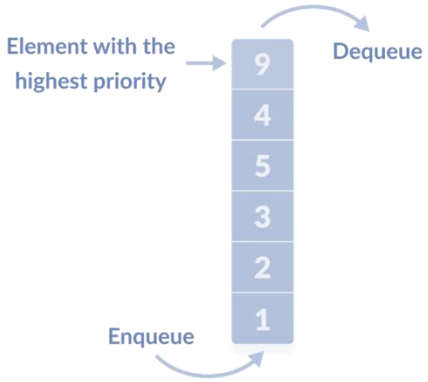
2.Circular Queue



Gambar 3 ilustrasi circular queue

Pada circular queue, simpul terakhir terhubung ke simpul pertama. Queue jenis ini juga dikenal sebagai Ring Buffer karena semua ujungnya terhubung ke ujung yang lain. Penyisipan terjadi di akhir antrian dan penghapusan di depan antrian.

3. Priotiy Queue



Gambar 4 Ilustrasi priority queue

Priority Queue adalah struktur data queue dimana simpul akan memiliki beberapa prioritas yang telah ditentukan. Simpul dengan prioritas terbesar akan menjadi yang pertama dihapus dari antrian. Sedangkan penyisipan item terjadi sesuai urutan kedatangannya.

4. Double-Ended Queue

Gambar 5 ilustrasi double-ended queue

Dalam double-ended queue (dequeue), operasi penyisipan dan penghapusan dapat terjadi di ujung depan dan belakang dari 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.

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

Berikut ini adalah beberapa fungsi queue yang paling umum dalam struktur data:

- Queue banyak digunakan untuk menangani lalu lintas (traffic) situs web.
- Membantu untuk mempertahankan playlist yang ada pada aplikasi media player
- Queue digunakan dalam sistem operasi untuk menangani interupsi.
- Membantu dalam melayani permintaan pada satu sumber daya bersama, seperti printer, penjadwalan tugas CPU, dll.

Kelebihan queue di antarnya:

- Data dalam jumlah besar dapat dikelola secara efisien.
- Operasi seperti penyisipan dan penghapusan dapat dilakukan dengan mudah karena mengikuti aturan masuk pertama keluar pertama.
- Queue berguna ketika layanan tertentu digunakan oleh banyak konsumen.
- Queue cepat untuk komunikasi antar-proses data.
- Queue dapat digunakan dalam implementasi struktur data lainnya.

Kelemahan struktur data queue adalah sebagai berikut:

- Operasi seperti penyisipan dan penghapusan elemen dari tengah cenderung banyak memakan waktu.
- Dalam queue konvensional, elemen baru hanya dapat dimasukkan ketika elemen yang ada dihapus dari antrian.
- Mencari elemen data pada struktur queue membutuhkan time complexity O(N).
- Ukuran maksimum antrian harus ditentukan sebelumnya.

IV. PROSEDUR PERCOBAAN

Adapun source code untuk percobaan ini adalah sebagai berikut :

4.1. Percobaan 6-1: Queue dengan Struct

```cpp
#include<iostream>
using namespace std;
#define MAX 8

typedef struct{
    int data[MAX];
    int head;
    int tail;
} Queue;
Queue antrian;

void create();
int isEmpty();
int isFull();
void enqueue(int data);
int dequeue();
void clear();
void tampil();

int main()
{
    int pil;
    int data;
    create();

    do {
        cout << "1. Enqueue" << endl;
        cout << "2. Dequeue" << endl;
        cout << "3. Tampil" << endl;
        cout << "4. Clear" << endl;
        cout << "5. Exit" << endl;
        cout << "Pilihan = ";
        cin >> pil;

        switch(pil){
            case 1:
                cout << "Data = ";
                cin >> data;
                enqueue(data);
                break;
            case 2:
                cout << "Elemen yang keluar : " << dequeue() << endl;
                break;
            case 3:
                tampil();
                break;
            case 4:
                clear();
                break;
        }
    } while(pil!=5);

    return 0;
```

```
}

void create()
{
    antrian.head=antrian.tail=-1;
}

int isEmpty()
{
    if(antrian.tail==-1)
    {
        return 1;
    }
    else
    {
    return 0;
    }
}

int isFull()
{
    if(antrian.tail==MAX-1)
    {
        return 1;
    }
    else
    {
    return 0;
    }
}

void enqueue(int data)
{
    if(isEmpty()==1)
    {
        antrian.head=antrian.tail=0;
        antrian.data[antrian.tail]=data;
        cout << antrian.data[antrian.tail] << " masuk!" << endl;
    } else if(isFull()==0)
    {
        antrian.tail++;
        antrian.data[antrian.tail]=data;
        cout << antrian.data[antrian.tail] << " masuk!" << endl;
    }
}

int dequeue()
{
    int i;
    int e = antrian.data[antrian.head];
    for(i=antrian.head;i<=antrian.tail-1;i++)
    {
        antrian.data[i] = antrian.data[i+1];
```

```
    }
    antrian.tail--;
    return e;
}

void clear()
{
    antrian.head=antrian.tail=-1;
    cout << "data clear" << endl;
}

void tampil()
{
    if(isEmpty()==0)
    {
        for(int i=antrian.head;i<=antrian.tail;i++)
        {
            cout << antrian.data[i] << " ";
        }
        cout << endl;
    }
    else
    {
        cout << "data kosong!" << endl;
    }
};
```

Gambar 4.1 Queue dengan Struct

4.2 Percobaan 6-2: Queue dengan Array Enqueue

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

int enqueue(int [], int);
void display(int [], int, int);
const int SIZE = 50;
int queue[SIZE];
int front=-1;
int rear=-1;

int main()
{
    int item, check;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan elemen pada antrian= ";
        cin>>item;
        check = enqueue(queue, item);
        if(check == -1)
        {
            cout<<"Overflow..!!..Keluar program.." << endl;
            exit(1);
        }
        cout<<"Antrian berhasil ditambahkan..." << endl;
        cout<<"Queue (Depan...ke...Belakang):" << endl;
        display(queue, front, rear);
        cout<<"Mau menambahkan antrian? (y/n) ";
        cin>>ch;
    }
    return 0;
}

int enqueue(int queue[], int elem)
{
    if(rear == SIZE-1)
    {
        return -1;
    }
    else if(rear == -1)
    {
        front = rear = 0;
        queue[rear] = elem;
    }
    else
    {
        rear++;
        queue[rear] = elem;
    }
    return 0;
}

void display(int queue[], int front, int rear)
{
    if(front == -1)
    {
        return;
    }
    for(int i=front; i<rear; i++)
    {
        cout<<queue[i]<<" <- ";
    }
    cout<<queue[rear]<< endl;
}
```

Gambar 4.2 Queue dengan Array Enqueue

4.3 Percobaan 6-3: Queue dengan Array Dequeue

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

int dequeue(int []);
int enqueue(int [], int);
void display(int [], int, int);
const int SIZE = 50;
int queue[SIZE];
int front=-1;
int rear=-1;

int main()
{
    int item, check;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan data pada antrian= ";
        cin>>item;
        check = enqueue(queue, item);
        if(check == -1)
        {
            cout<<"Overflow..!!..Keluar program.." << endl;
            exit(1);
        }
        cout<<"Antrian berhasil ditambahkan..." << endl;
        cout<<"Queue (Depan...ke...Belakang):" << endl;
        display(queue, front, rear);
        cout<<"Mau menambahkan antrian? (y/n) ";
        cin>>ch;
    }
    cout<<"Memulai pengeluaran elemen...\n";
    ch='y';
    while(ch=='y' || ch=='Y')
    {
        check = dequeue(queue);
        if(check == -1)
        {
            cout<<"Underflow..!!..Keluar program.." << endl;
            exit(2);
        }
        else
        {
            cout<<"Elemen yang dikeluarkan dari antrian: "<<check<< endl;
            cout<<"Queue (Depan...ke...Belakang):" << endl;
            display(queue, front, rear);
        }
        cout<<"Mau mengeluarkan elemen dari antrian lagi? (y/n) ";
        cin>>ch;
    }
    return 0;
}

```

```
54  int enqueue(int queue[], int elem)
55  {
56      if(rear == SIZE-1)
57      {
58          return -1;
59      }
60      else if(rear == -1)
61      {
62          front = rear = 0;
63          queue[rear] = elem;
64      }
65      else
66      {
67          rear++;
68          queue[rear] = elem;
69      }
70      return 0;
71  }
72
73  int dequeue(int queue[])
74  {
75      int retn;
76      if(front == -1)
77      {
78          return -1;
79      }
80      else
81      {
82          retn = queue[front];
83          if(front == rear)
84          {
85              front = rear = -1;
86          }
87          else
88          {
89          front++;
90          }
91      }
92      return retn;
93  }
94
95  void display(int queue[], int front, int rear)
96  {
97      if(front == -1)
98      {
99          return;
100     }
101     for(int i=front; i<rear; i++)
102     {
103         cout<<queue[i]<<" <- ";
104     }
105     cout<<queue[rear]<< endl;
106 }
```

Gambar 4.3 Queue dengan Array Dequeue

4.4 Percobaan 6-4: Queue dengan Linked List Enqueue

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

struct node
{
    int info;
    node *next;
} *front, *newptr, *save, *ptr, *rear;

node *create_new_node(int);
void insert(node *);
void display(node *);

int main()
{
    front = rear = NULL;
    int inf;
    int count=0;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan data pada antrian= ";
        cin>>inf;
        newptr = create_new_node(inf);
        if(newptr == NULL)
        {
            cout<<"Tidak dapat membuat node..!!..Keluar program.." << endl;
            exit(1);
        }
        insert(newptr);
        cout<<"Antrian berhasil ditambahkan..." << endl;
        cout<<"Queue (Depan...ke...Belakang):" << endl;
        display(front);
        cout<<"Mau menambahkan antrian? (y/n) ";
        cin>>ch;
    }
    return 0;
}

node *create_new_node(int x)
{
    ptr = new node;
    ptr->info = x;
    ptr->next = NULL;
    return ptr;
}
```

```
47
48  void insert(node *n)
49  {
50      if(front == NULL)
51      {
52          front = rear = n;
53      }
54      else
55      {
56          rear->next = n;
57          rear = n;
58      }
59  }
60
61  void display(node *n)
62  {
63      while(n != NULL)
64      {
65          cout<<n->info<<" -> ";
66          n = n->next;
67      }
68      cout<<"!!\n";
69  }
```

Gambar 4.4 Queue dengan Linked List Enqueue

4.5 Percobaan 6-5: Queue dengan Linked List Dequeue

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

struct node
{
    int info;
    node *next;
} *front, *newptr, *save, *ptr, *rear;

node *create_new_node(int);
void insert(node *);
void delete_node_queue();
void display(node *);

int main()
{
    front = rear = NULL;
    int inf;
    int count=0;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan data pada antrian= ";
        cin>>inf;
        newptr = create_new_node(inf);
        if(newptr == NULL)
        {
            cout<<"Tidak dapat membuat node..!!..Keluar program.." << endl;
            exit(1);
        }
        insert(newptr);
        cout<<"Antrian berhasil ditambahkan..." << endl;
        cout<<"Queue (Depan...ke...Belakang):" << endl;
        display(front);
        cout<<"Mau menambahkan antrian? (y/n) ";
        cin>>ch;
    }

    do {
        cout<<"Queue (Depan...ke...Belakang):" << endl;
        display(front);
        if(count == 0)
        {
            cout<<"Mau mengeluarkan elemen dari antrian? (y/n) ";
            count++;
        }
        else
        {
            cout<<"Mau mengeluarkan elemen dari antrian lagi? (y/n) ";
        }
        cin>>ch;
        if(ch=='y' || ch=='Y')
        {
```

```
            delete_node_queue();
        }
        cout << endl;
    } while(ch=='y' || ch=='Y');
    return 0;
}

node *create_new_node(int x)
{
    ptr = new node;
    ptr->info = x;
    ptr->next = NULL;
    return ptr;
}

void insert(node *n)
{
    if(front == NULL)
    {
        front = rear = n;
    }
    else
    {
        rear->next = n;
        rear = n;
    }
}

void delete_node_queue()
{
    if(front == NULL)
    {
        cout<<"Overflow..!!..Keluar program.." << endl;
        exit(2);
    }
    else
    {
        ptr = front;
        front = front->next;
    }
}

void display(node *n)
{
    while(n != NULL)
    {
        cout<<n->info<<" -> ";
        n = n->next;
    }
    cout<<"!!\n";
}
```

Gambar 4.5 Queue dengan Linked List Dequeue

V. PEMBAHASAN

Adapun source code percobaan ini adalah sebagai berikut :

5.1. Percobaan 6-1: Queue dengan Struct

5.1.a Source Code Percobaan 6-1: Queue dengan Struct

```cpp
#include<iostream>
using namespace std;
#define MAX 8

typedef struct{
    int data[MAX];
    int head;
    int tail;
} Queue;
Queue antrian;

void create();
int isEmpty();
int isFull();
void enqueue(int data);
int dequeue();
void clear();
void tampil();

int main()
{
    int pil;
    int data;
    create();

    do {
        cout << "1. Enqueue" << endl;
        cout << "2. Dequeue" << endl;
        cout << "3. Tampil" << endl;
        cout << "4. Clear" << endl;
        cout << "5. Exit" << endl;
        cout << "Pilihan = ";
        cin >> pil;

        switch(pil){
            case 1:
                cout << "Data = ";
                cin >> data;
                enqueue(data);
                break;
            case 2:
                cout << "Elemen yang keluar : " << dequeue() << endl;
                break;
            case 3:
                tampil();
                break;
            case 4:
                clear();
                break;
        }
    } while(pil!=5);

    return 0;
```

```
}

void create()
{
    antrian.head=antrian.tail=-1;
}

int isEmpty()
{
    if(antrian.tail==-1)
    {
        return 1;
    }
    else
    {
    return 0;
    }
}

int isFull()
{
    if(antrian.tail==MAX-1)
    {
        return 1;
    }
    else
    {
    return 0;
    }
}

void enqueue(int data)
{
    if(isEmpty()==1)
    {
        antrian.head=antrian.tail=0;
        antrian.data[antrian.tail]=data;
        cout << antrian.data[antrian.tail] << " masuk!" << endl;
    } else if(isFull()==0)
    {
        antrian.tail++;
        antrian.data[antrian.tail]=data;
        cout << antrian.data[antrian.tail] << " masuk!" << endl;
    }
}

int dequeue()
{
    int i;
    int e = antrian.data[antrian.head];
    for(i=antrian.head;i<=antrian.tail-1;i++)
    {
        antrian.data[i] = antrian.data[i+1];
```

```
107      }
108      antrian.tail--;
109      return e;
110  }
111
112  void clear()
113  {
114      antrian.head=antrian.tail=-1;
115      cout << "data clear" << endl;
116  }
117
118  void tampil()
119  {
120      if(isEmpty()==0)
121      {
122          for(int i=antrian.head;i<=antrian.tail;i++)
123          {
124              cout << antrian.data[i] << " ";
125          }
126          cout << endl;
127      }
128      else
129      {
130          cout << "data kosong!" << endl;
131      }
132  };
```

Gambar 5.1.a Source Code Queue dengan Struct

Berdasarkan gambar 5.1.a yang merupakan source code dari program queue dengan struct, dapat dilihat pada baris ke 1 terdapat penggunaan library iostream yang berfungsi untuk input dan output dari user. Pada baris ke 2 terdapat penggunaan penamaan standar bagi compiler. Pada baris ke 3 terdapat deklarasi variabel global MAX dengan nilai 8. Pada baris ke 5 terdapat pendefinisian alias dari struct menjadi Queue. Di dalam queue, pada baris ke 6 terdapat deklarasi array integer data dengan panjang array mengikuti nilai variabel MAX. Pada baris ke 7 terdapat deklarasi variabel integer head. Pada baris ke 8 terdapat inisialisasi variabel integer tail. Pada baris ke 10 terdapat inisialisasi penamaan Queue menjadi antrian. Pada baris ke 12 terdapat deklarasi fungsi void create yang berfungsi untuk membuat elemen data.

Pada baris ke 13 terdapat deklarasi fungsi integer isEmpty yang berfungsi untuk memastikan apakah antrian kosong atau tidak. Pada baris ke 14 terdapat deklarasi fungsi integer isFull yang berfungsi untuk memastikan apakah antrian penuh atau tidak. Pada baris ke 15 terdapat deklarasi fungsi void enqueue dengan parameter integer data yang berfungsi untuk memasukkan elemen data pada antrian. Pada baris ke 16 terdapat deklarasi fungsi integer dequeue yang berfungsi untuk mengeluarkan elemen data pada antrian. Pada baris ke 17 terdapat deklarasi fungsi void clear yang berfungsi untuk mengosongkan antrian. Pada baris ke 18 terdapat deklarasi fungsi void tampil yang berfungsi untuk menampilkan antrian yang ada ke terminal. Pada baris ke 20 terdapat inisialisasi fungsi integer main. Di dalamnya, pada baris ke 22 terdapat deklarasi fungsi integer pil. Pada baris ke 23 terdapat deklarasi fungsi integer data. Pada baris ke 24 terdapat perintah untuk menjalankan fungsi create. Pada baris ke 26 terdapat inisialisasi do dari perulangan while-do. Pada baris ke 27 terdapat perintah untuk menampilkan pesan "1. Enqueue". Pada baris ke 28 terdapat perintah untuk menampilkan pesan "2. Denqueue". Pada baris ke 29 terdapat perintah untuk menampilkan pesan "3. Tampil". Pada baris ke 30 terdapat perintah untuk menampilkan pesan "4. Clear". Pada baris ke 31 terdapat perintah untuk menampilkan pesan "5. Exit". Pada baris ke 32 terdapat perintah untuk menampilkan pesan "Pilihan =" dan diikuti dengan perintah agar user untuk memasukkan nilai ke variabel pil. Nilai yang diinputkan user nantinya akan menjadi patokan bagi komputer untuk menjalankan menu yang dipilih. Pada baris ke 35 terdapat percabangan switch dengan kondisi dari nilai variabel pil. Pada baris ke 36 terdapat case 1 dimana program akan meminta user untuk memasukkan daya dan kemudian akan menjalankan fungsi enqueue dengan argumen data tersebut. Case ini akan menambahkan data ke dalam antrian. Pada baris ke 41 terdapat case 2, dimana program akan menampilkan pesan "Elemen yang keluar : " dan menjalankan fungsi dequeue. Case ini akan mengeluarkan data dari antrian. Pada baris ke 44 terdapat case 3 dimana program akan menjalankan fungsi tampil. Case ini berfungsi untuk menampilkan antrian yang ada saat fungsi tersebut dipanggil. Pada baris ke 47 terdapat case 4 dimana program akan menjalankan fungsi clear. Case ini berfungsi untuk mengosongkan antrian yang ada. [ada baris ke 51 terdapat while dari perulangan do-while dengan kondisi selama nilai pil != 5 maka ulangi

perulangan tersebut. Pada baris ke 56 terdapat inisialisasi fungsi void create. Di dalamnya, yaitu pada baris ke 58 terdapat inisialisasi nilai antrian.head = antrian.tail = -1 yang menandakan bahwa antrian kosong. Pada baris ke 61 terdapat inisialisasi nilai dari fungsi void isEmpty . Di dalamnya, terdapat percabangan if dengan kondisi jika antrian.tail = -1 yang berarti antrian kosong, maka kembalikan nilai -1. Jika tidak maka kembalikan nilai 0. Pada baris ke 73 terdapat inisialisasi fungsi void isFull. Di dalamnya terdapat percabangan if dengan kondisi jika antrian.tail = MAX-1 yang berarti antrian penuh, maka kembalikan nilai 1. Jika tidak maka kembalikan nilai 0. Pada baris ke 85 terdapat inisialisasi fungsi void enqueue dengan parameter integer data. Di dalamnya terdapat percabangan if dengan kondisi jika nilai dari menjalankan fungsi isEmpty = 1 maka inisialisasi nilai antrian.head = antrian.tail = 0, inisialisasi nilai antrian.data[antrian.tail] = data, dan tampilkan pesan bahwa data telah masuk. Jika tidak, maka jika nilai dari menjalankan fungsi isFull = 0 maka lakukan increment pada nilai antrian.tail, inisialisasi nilai antrian.data[antrian.head] = data dan tampilkan pesan bahwa data telah masuk. Pada baris ke 100 terdapat inisialisasi fungsi integer dequeue. Di dalamnya terdapat deklarasi variabel integer i, inisialisasi variabel integer e dengan nilai dari arntrian.data[antrian.head], dan perulangan for dengan inisialisasi i = antrian.head dan selama i ≤ antrian.tail – 1 maka lakukan increment pada nilai i dan lakukan inisialisasi nilai antrian.data pada indeks ke i dengan nilai dari antrian.data indeks ke i+1. Selain itu juga terdapat decrement nilai dari antrian .tail pada baris ke 108 dan fungsi akan mengembalikan nilai dari variabel e. Pada baris ke 112 terdapat inisialisasi dari fungsi void clear. Di dalamnya terdapat inisialisasi antrian.head = antrian.tail = -1 dan perintah untuk menampilkan pesan “data clear”. Pada baris ke 118 terdapat inisialisasi fungsi void tampil. Di dalamnya terdapat percabangan if dengan kondisi jika nilai dari menjalankan fungsi isEmpty = 0 maka lakukan perulangan for dengan inisialisasi nilai integer i = antrian.head dan selama nilai i ≤ antrian.tail maka lakukan increment pada nilai i dan tampilkan nilai dari antrian.data indeks ke i. Kemudian jika isEmpty bukan bernilai 0 maka tampilkan pesan “data kosong”.

5.1.b Output Percobaan 6-1: Queue dengan Struct

```
6-1queuestruct } ; if ($?) { .\6-1queuestruct }
1. Enqueue
2. Dequeue
3. Tampil
4. Clear
5. Exit
Pilihan = 1
Data = 12
12 masuk!
1. Enqueue
2. Dequeue
3. Tampil
4. Clear
5. Exit
Pilihan = 1
Data = 90
90 masuk!
1. Enqueue
2. Dequeue
3. Tampil
4. Clear
5. Exit
Pilihan = 1
Data = 34
34 masuk!
1. Enqueue
2. Dequeue
3. Tampil
4. Clear
5. Exit
Pilihan = 2
Elemen yang keluar : 12
1. Enqueue
2. Dequeue
3. Tampil
4. Clear
5. Exit
Pilihan = 3
90 34
1. Enqueue
2. Dequeue
3. Tampil
4. Clear
5. Exit
Pilihan = 4
data clear
1. Enqueue
2. Dequeue
3. Tampil
4. Clear
5. Exit
Pilihan = 5
```

Gambar 5.1.b Output Queue dengan Struct

Berdasarkan gambar 5.1.b yang merupakan output dari queue dengan struct, dapat dilihat bahwa pertama program menampilkan pilihan opsi menu yang ada, kemudian program meminta user untuk memasukkan pilihan. Ini sesuai dengan perintah pada baris ke 27 – 33. Kemudian user memasukkan pilihan 1 yang berarti melakukan enqueue terhadap antrian. Kemudian karena user memilih opsi 1, program berpindah ke case 1, dimana program meminta user untuk memasukkan data yang akan dienqueue. Kemudian user memasukkan nilai 12. Kemudian program memasukkan data tersebut dengan cara memanggil fungsi enqueue dengan argument data yang sebelumya diinputkan oleh user. Kemudian program kembali menampilkan pilihan opsi menu yang ada, dan kemudian program meminta user untuk memasukkan pilihan. Kemudian user kembali memasukkan pilihan 1 dan menambahkan data 90 dan 34, sehingga pada sekarang terdapat 3 data pada antrian. Kemudian program kembali menampilkan pilihan opsi menu yang ada, dan kemudian program meminta user untuk memasukkan pilihan. Berikutnya, user memilih opsi 2 yang berarti melakukan dequeue terhadap antrian. Karena user memilih opsi 2, maka program beralih ke case 2 dimana progam menjalankan fungsi dequeue dan menampilkan returnnya, yaitu data yang dihapuskan dari antrian. Karena pada queue data yang pertama kali dimasukkan adalah yang pertama kali juga dikeluarkan dan data yang pertama dimasukkan adalah 12, maka data 12 lah yang dikeluarkan dari antrian yang ada. Kemudian program kembali menampilkan pilihan opsi menu yang ada, dan kemudian program meminta user untuk memasukkan pilihan. Berikutnya user memasukkan opsi 3, yaitu opsi untuk menampilkan antrian yang ada. Karena user memilih opsi 3, maka program beralih ke case 3, dimana program menjalankang fungsi tampil sehingga program menampilkan seluruh data pada antrian yang ada. Dapat dilihat bahwa saat opsi tersebut dipilih hanya terdapat dua data pada antrian, yaitu 90 dan 34. Kemudian program kembali menampilkan pilihan opsi menu yang ada, dan kemudian program meminta user untuk memasukkan pilihan. Berikutinya user memilih opsi 4 yaitu opsi untuk mengosongkan antrian. Karena user memilih opsi 4, maka program menjalankan fungsi clear yang mana akan membuat bagian belakang dari antrian yang tadinya berada pada indeks 1 menjadi -1 sehingga seolah-olah antrian tersebut kosong. Kemudian program kembali menampilkan pilihan opsi menu yang ada, dan

kemudian program meminta user untuk memasukkan pilihan. Berikutinya user memilih opsi 5 yaitu opsi untuk keluar dari program. Karena user memasukkan nilai 5, maka nilai tersebut membuat kondisi perulangan yang diinisialisasikan pada baris ke 51 menjadi bernilai false, sehingga program tidak melakukan perulangan kembali dan program berhenti pada return 0.

5.2 Percobaan 6-2: Queue dengan Array Enqueue

5.2.a Source code Percobaan 6-2: Queue dengan Array Enqueue

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

int enqueue(int [], int);
void display(int [], int, int);
const int SIZE = 50;
int queue[SIZE];
int front=-1;
int rear=-1;

int main()
{
    int item, check;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan elemen pada antrian= ";
        cin>>item;
        check = enqueue(queue, item);
        if(check == -1)
        {
            cout<<"Overflow..!!..Keluar program.." << endl;
            exit(1);
        }
        cout<<"Antrian berhasil ditambahkan..." << endl;
        cout<<"Queue (Depan...ke...Belakang):" << endl;
        display(queue, front, rear);
        cout<<"Mau menambahkan antrian? (y/n) ";
        cin>>ch;
    }
    return 0;
}

int enqueue(int queue[], int elem)
{
    if(rear == SIZE-1)
    {
        return -1;
    }
    else if(rear == -1)
    {
        front = rear = 0;
        queue[rear] = elem;
    }
    else
    {
        rear++;
        queue[rear] = elem;
    }
    return 0;
}

void display(int queue[], int front, int rear)
{
    if(front == -1)
    {
        return;
    }
    for(int i=front; i<rear; i++)
    {
        cout<<queue[i]<<" <- ";
    }
    cout<<queue[rear]<< endl;
}
```

Gambar 5.2.a Source Code Queue dengan Array Enqueue

Berdasarkan gambar 5.2.a yang merupakan source code dari program queue dengan array enqueue dapat dilihat pada baris ke 1 terdapat penggunaan library iostream yang berfungsi untuk input dan output dari user. Pada baris ke 2 terdapat penggunaan penamaan standar bagi compiler. Pada baris ke 4 terdapat deklarasi fungsi integer enqueue dengan parameter array integer dan variabel integer. Pada bari s ke 5 terdapat deklarasi fungsi void display dengan parameter array integer, variabel integer, dan variabel integer. Pada baris ke 6 terdapat inisialisasi variabel konstan integer dengan nama SIZE dan nilai 50. Pada baris ke 7 terdapat deklarasi array queue dengan panjang array dari nilai variabel SIZE. Pada baris ke 8 terdapat inisialisasi variabel integer front dengan nilai -1. Pada baris ke 9 terdapat inisialisasi variabel integer rear dengan nilai -1. Pada baris ke 11 terdapat inisialisasi fungsi integer main. Di dalamnya, pada baris ke 13 terdapat deklarasi variabel integer item dan check. Pada baris ke 14 terdapat inisialisasi variabel char ch dengan nilai y. Pada baris ke 15 terdapat perulangan while dengan kondisi ch = y/Y. pada baris ke 17, di dalam perulangan, terdapat perintah untuk menampilkan pesan "Masukkan elemen pada antrian= ". Pada baris ke 18 terdapat perintah agar user untuk memasukkan nilai ke dalam variabel item. Nilai ini nantinya menjadi data yang akan dimasukkan ke dalam antrian. Kemudian pada baris ke 19 terdapat inisialisasi variabel check dengan nilai dari hasil menjalankan fungsi enqueue dengan argumen queue dan item. Pada baris ke 20 terdapat percabangan if dengan kondisi jika nilai check = -1 maka tampilkan pesan "Overflow..!!..Keluar program.." dan exit(1). Pada baris ke 25 terdapat perintah untuk menampilkan pesan "Antrian berhasil ditambahkan". Pada baris ke 26 terdapat perintah untuk menampilkan pesan "Queue (Depan…ke…belakang)". Pada baris ke 27 terdapat perintah untuk menjalankan fungsi display dengan argumen queue, front, dan rear. Pada baris ke 28 terdapat perintah untuk menampilkan pesan "Mau menambahkan antrian? (y/n)". Pada baris ke 29 terdapat perintah agar user untuk memasukkan nilai ke dalam variabel ch. Kemudian pada baris ke 31 terdapat return 0 yang berarti program berjalan dengan normal dan tidak ada kendala. Pada baris ke 34 terdapat inisialisasi fungsi integer enqueue yang berfungsi untuk menambahkan data ke dalam antrian dengan parameter array integer queue dan variabel integer elem. Pada baris ke 36, di dalam fungsi, terdapat percabangan if dengan kondisi jika rear =

SIZE-1 maka kembalikan nilai -1. Jika tidak, maka jika rear = -1, maka inisialisasi nilai front = rear = 0 dan inisialisasi nilai queue[rear] = elem lalu keluar dari percabangan dan kembalikan nilai 0. Jika tidak, maka increment nilai variabel rear dan inisialisasi nilai queue[rear] = elem lalu keluar dari percabangan dan kembalikan nilai 0. Kemudian pada baris ke 53 terdapat inisialisasi fungsi void display yang berfungsi untuk menampilkan antrian yang ada dengan parameter array integer queue, variabel integer front, dan variabel integer rear. Pada baris ke 55, di dalam fungsi, terdapat percabangan if dengan kondisi jika nilai front = -1, maka jangan kembalikan return apapun. Selain itu terdapat juga perulangan for dengan inisialisasi integer i = front dan jika i < rear maka increment nilai i tampilkan nilai dari queue indeks ke i. Pada baris ke 63, masih di dalam fungsi, terdapat perintah untuk menampilkan nilai dari queue[rear].

5.2.b Output Percobaan 6-2: Queue dengan Array Enqueue

```
-o 6-2queueArrayEnqueue } ; if ($?) { .\6-2queueArrayEnqueue }
Masukkan elemen pada antrian= 12
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12
Mau menambahkan antrian? (y/n) y
Masukkan elemen pada antrian= 90
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12 <- 90
Mau menambahkan antrian? (y/n) y
Masukkan elemen pada antrian= 34
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12 <- 90 <- 34
Mau menambahkan antrian? (y/n) n
```

Gambar 5.2.b Output Queue dengan Array Enqueue

Berdasarkan percobaan 5.2.b yang merupakan output dari queue dengan array enqueue, dapat dilihat bahwa pertama program menampilkan pesan “Masukkan elemen pada antrian” dan meminta user untuk memasukkan nilai ke variabel item. Kemudian user memasukkan nilai 12. Nilai yang dimasukkan user ini nantinya akan dijadikan sebagai argumen dalam memanggil fungsi enqueue dan kemudian akan dimasukkan ke dalam antrian oleh fungsi tersebut. Kemudian karena pada saat pemanggilan fungsi enqueue, nilai variabel rear ≠ size - 1 maka fungsi enqueue berpindah ke percabangan else if dimana rear = -1 kemudian program menginisialisasi nilai front = rear = 0 dan memasukkan data yang dimasukkan oleh user ke dalam antrian. Kemudian program mengembalikan nilai 0. Karena pada percabangan baris ke-20 nilai check tidak bernilai -1 maka percabangan bernilai false dan program tidak overflow. Kemudian program menampilkan pesan “Antrian berhasil ditambahkan” dan pesan “Queue (depan…ke…belakang):”. Kkemudian program menjalankan fungsi display untuk menampilkan antrian yang ada saat fungsi tersebut dipanggil. Kemudian program menampilkan pesan “Mau menambahkan antrian? (y/n)” dan meminta input user. Input user ini nantinya akan dijadikan acuan bagi program untuk memilih apakah akan mengulangi memasukkan elemen atau mengakhiri program. Kemudian user memasukkan nilai y yang berarti akan kembali memasukkan elemen ke dalam antrian. Karena user

masukan nilai y maka perulangan bernilai true dan mengulang kembali meminta user untuk memasukkan elemen kepada antrian. Berikutnya user memasukkan nilai 90. Kemudian karena antrian masih belum penuh maka program kembali menampilkan "Antrian berhasil ditambahkan" dan pesan "Queue (depan…ke…belakang):" serta menampilkan antrian yang ada dengan memanggil fungsi display. Kemudian program kembali menanyakan kepada user apakah mau menambahkan antrian dan meminta user untuk memasukkan input. Kemudian user kembali memasukkan nilai y yang berarti user ingin kembali memasukkan elemen pada antrian. Karena user masukan nilai y maka perulangan bernilai true dan mengulang kembali meminta user untuk memasukkan elemen kepada antrian. Kemudian user memasukkan nilai 34 dan karena antrian masih belum penuh maka percabangan pada baris ke-20 masih bernilai false dan program tidak termasuk overflow. Kemudian program kembali menampilkan pesan Antrian berhasil ditambahkan" dan pesan "Queue (depan…ke…belakang):" dan kembali menampilkan antrian yang tersedia saat fungsi display dipanggil. Kemudian program kembali menanyakan kepada user apakah ingin menambahkan elemen pada antrian kemudian user memasukkan nilai n yang berarti user tidak ingin lagi memasukkan elemen pada antrian. Karena user memasukkan nilai n maka kondisi perulangan untuk masukkan elemen yang ada pada baris ke-15 bernilai false dan perulangan untuk memasukkan elemen pada antrian tidak dijalankan kembali oleh program. Kemudian program berakhir dan mengembalikan nilai 0 yang berarti program berjalan dengan normal.

5.3 Percobaan 6-3: Queue dengan Array Dequeue

5.3.a Source Code Percobaan 6-3: Queue dengan Array Dequeue

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

int dequeue(int []);
int enqueue(int [], int);
void display(int [], int, int);
const int SIZE = 50;
int queue[SIZE];
int front=-1;
int rear=-1;

int main()
{
    int item, check;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan data pada antrian= ";
        cin>>item;
        check = enqueue(queue, item);
        if(check == -1)
        {
            cout<<"Overflow..!!..Keluar program.." << endl;
            exit(1);
        }
        cout<<"Antrian berhasil ditambahkan..." << endl;
        cout<<"Queue (Depan...ke...Belakang):" << endl;
        display(queue, front, rear);
        cout<<"Mau menambahkan antrian? (y/n) ";
        cin>>ch;
    }
    cout<<"Memulai pengeluaran elemen...\n";
    ch='y';
    while(ch=='y' || ch=='Y')
    {
        check = dequeue(queue);
        if(check == -1)
        {
            cout<<"Underflow..!!..Keluar program.." << endl;
            exit(2);
        }
        else
        {
            cout<<"Elemen yang dikeluarkan dari antrian: "<<check<< endl;
            cout<<"Queue (Depan...ke...Belakang):" << endl;
            display(queue, front, rear);
        }
        cout<<"Mau mengeluarkan elemen dari antrian lagi? (y/n) ";
        cin>>ch;
    }
    return 0;
}

```

```
54  int enqueue(int queue[], int elem)
55  {
56      if(rear == SIZE-1)
57      {
58          return -1;
59      }
60      else if(rear == -1)
61      {
62          front = rear = 0;
63          queue[rear] = elem;
64      }
65      else
66      {
67          rear++;
68          queue[rear] = elem;
69      }
70      return 0;
71  }
72
73  int dequeue(int queue[])
74  {
75      int retn;
76      if(front == -1)
77      {
78          return -1;
79      }
80      else
81      {
82          retn = queue[front];
83          if(front == rear)
84          {
85              front = rear = -1;
86          }
87          else
88          {
89          front++;
90          }
91      }
92      return retn;
93  }
94
95  void display(int queue[], int front, int rear)
96  {
97      if(front == -1)
98      {
99          return;
100     }
101     for(int i=front; i<rear; i++)
102     {
103         cout<<queue[i]<<" <- ";
104     }
105     cout<<queue[rear]<< endl;
106 }
```

Gambar 5.3.a Source Code Queue dengan Array Dequeue

Berdasarkan gambar 5.3.a yang merupakan source code dari program queue dengan array dequeue, dapat dilihat pada baris ke 1 terdapat penggunaan library iostream yang berfungsi untuk input dan output dari user. Pada baris ke 2 terdapat penggunaan penamaan standar bagi compiler. Pada baris ke 4 terdapat deklarasi fungsi dequeue dengan parameter array integer. Pada baris ke 5 terdapat deklarasi fungsi integer enqueue dengan parameter array integer dan variabel integer. Pada baris ke 6 terdapat deklarasi fungsi void display dengan parameter array integer, variabel integer, dan variabel integer. Pada baris ke 7 terdapat inisialisasi variabel konstan integer dengan nama SIZE dan nilai 50. Pada baris ke 8 terdapat deklarasi array queue dengan panjang array dari nilai variabel SIZE. Pada baris ke 9 terdapat inisialisasi variabel integer front dengan nilai -1. Pada baris ke 10 terdapat inisialisasi variabel integer rear dengan nilai -1. Pada baris ke 12 terdapat inisialisasi fungsi integer main. Di dalamnya, pada baris ke 14 terdapat deklarasi variabel integer item dan check. Pada baris ke 15 terdapat inisialisasi variabel char ch dengan nilai y. Pada baris ke 16 terdapat perulangan while dengan kondisi ch = y/Y. pada baris ke 18, di dalam perulangan, terdapat perintah untuk menampilkan pesan "Masukkan elemen pada antrian= ". Pada baris ke 19 terdapat perintah agar user untuk memasukkan nilai ke dalam variabel item. Nilai ini nantinya menjadi data yang akan dimasukkan ke dalam antrian. Kemudian pada baris ke 20 terdapat inisialisasi variabel check dengan nilai dari hasil menjalankan fungsi enqueue dengan argumen queue dan item. Pada baris ke 21 terdapat percabangan if dengan kondisi jika nilai check = -1 maka tampilkan pesan "Overflow..!!..Keluar program.." dan exit(1). Pada baris ke 26 terdapat perintah untuk menampilkan pesan "Antrian berhasil ditambahkan". Pada baris ke 27 terdapat perintah untuk menampilkan pesan "Queue (Depan…ke…belakang)". Pada baris ke 28 terdapat perintah untuk menjalankan fungsi display dengan argumen queue, front, dan rear. Pada baris ke 29 terdapat perintah untuk menampilkan pesan "Mau menambahkan antrian? (y/n)". Pada baris ke 30 terdapat perintah agar user untuk memasukkan nilai ke dalam variabel ch. Pada baris ke 32 terdapat perintah untuk menampilkan pesan "Memulai pengeluaran elemen…". Pada baris ke 33 terdapat inisialisasi nilai variabel ch = y. Pada baris ke 34 terdapat perulangan while dengan kondisi ch = y atau Y. Pada baris ke 36 terdapat inisialisasi variabel check dengan hasil dari

menjalankan fungsi dequeue dengan argumen queue. Pada baris ke 37 terdapat percabangan if dengan kondisi jika nilai variabel check sama dengan -1 maka tampilkan pesan "Underflow..!!..Keluar program.." dan exit(2). Kemudian jika check tidak sama dengan -1 maka tampilkan pesan “Elemen yang dikeluarkan dari antrian :” dan nilai dari variabel check, pesan "Queue (Depan...ke...Belakang):", dan memanggil fungsi display dengan argumen queue, front, dan rear. Kemudian, keluar dari percabangan, terdapat perintah untuk menampilkan pesan “Mau mengeluarkan elemen dari antrian lagi? (y/n)” dan kemudian terdapat perintah agar user untuk memasukkan nilai ke variabel ch. Input dari user ini nantinya akan dijadikan acuan bagi program apakah ingin kembali mengeluarkan elemen dari antrian lagi atau tidak. Kemudian pada baris ke 51 terdapat return 0 yang berarti program berjalan dengan normal dan tidak ada kendala. Pada baris ke 54 terdapat inisialisasi fungsi integer enqueue yang berfungsi untuk menambahkan data ke dalam antrian dengan parameter array integer queue, dan variabel integer elem. Pada baris ke 56, di dalam fungsi, terdapat percabangan if dengan kondisi jika rear = SIZE-1 maka kembalikan nilai -1. Jika tidak, maka jika rear = -1, maka inisialisasi nilai front = rear = 0 dan inisialisasi nilai queue[rear] = elem lalu keluar dari percabangan dan kembalikan nilai 0. Jika tidak, maka increment nilai variabel rear dan inisialisasi nilai queue[rear] = elem lalu keluar dari percabangan dan kembalikan nilai 0. Pada baris ke- 3 terdapat inisialisasi dari fungsi integer dequeue yang berfungsi untuk menghapus data dari array dengan parameter array integer queue. Di dalamnya, pada baris ke 75, terdapat deklarasi variabel integer retn. Pada baris ke 76 terdapat percabangan if dengan kondisi jika nilai dari variabel front = -1 maka kembalikan nilai-1 dan jika tidak maka inisialisasi nilai retn = queue[front]. Pada baris ke 83, di dalam percabangan else, terdapat percabangan if (nested) dengan kondisi jika nilai front = rear maka inisialisasi nilai front = rear = -1 jika tidak maka lakukan increment terhadap nilai variabel front kemudian keluar dari percabangan dan kembalikan nilai retn. Kemudian pada baris ke 95 terdapat inisialisasi fungsi void display yang berfungsi untuk menampilkan antrian yang ada saat itu dengan parameter array integer queue, variabel integer front, dan variabel integer rear. Pada baris ke 97, di dalam fungsi, terdapat percabangan if dengan kondisi jika nilai front = -1, maka jangan kembalikan return apa pun. Selain itu

terdapat juga perulangan for dengan inisialisasi integer i = front dan jika i < rear maka increment nilai i tampilkan nilai dari queue indeks ke i. Pada baris ke 105, masih di dalam fungsi, terdapat perintah untuk menampilkan nilai dari queue[rear].

5.3.b Output Percobaan 6-3: Queue dengan Array Dequeue

```
-o 6-3queueArrayDequeue } ; if ($?) { .\6-3queueArrayDequeue }
Masukkan data pada antrian= 12
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12
Mau menambahkan antrian? (y/n) y
Masukkan data pada antrian= 90
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12 <- 90
Mau menambahkan antrian? (y/n) y
Masukkan data pada antrian= 34
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12 <- 90 <- 34
Mau menambahkan antrian? (y/n) n
Memulai pengeluaran elemen...
Elemen yang dikeluarkan dari antrian: 12
Queue (Depan...ke...Belakang):
90 <- 34
Mau mengeluarkan elemen dari antrian lagi? (y/n) y
Elemen yang dikeluarkan dari antrian: 90
Queue (Depan...ke...Belakang):
34
Mau mengeluarkan elemen dari antrian lagi? (y/n) y
Elemen yang dikeluarkan dari antrian: 34
Queue (Depan...ke...Belakang):
Mau mengeluarkan elemen dari antrian lagi? (y/n) y
Underflow..!!..Keluar program..
```

Gambar 5.3.b Output Queue dengan Array Dequeue

Berdasarkan percobaan 5.3.b yang merupakan output dari program queue dengan array dequeue, dapat dilihat bahwa pertama program menampilkan pesan "Masukkan elemen pada antrian" dan meminta user untuk memasukkan nilai ke variabel item. Kemudian user memasukkan nilai 12. Nilai yang dimasukkan user ini nantinya akan dijadikan sebagai argumen dalam memanggil fungsi enqueue dan kemudian akan dimasukkan ke dalam antrian oleh fungsi tersebut. Kemudian karena pada saat pemanggilan fungsi enqueue, nilai variabel rear ≠ size - 1 maka fungsi enqueue berpindah ke percabangan else if dimana rear = -1 kemudian program menginisialisasi nilai front = rear = 0 dan memasukkan data yang dimasukkan oleh user ke dalam antrian. Kemudian program mengembalikan nilai 0. Karena pada percabangan baris ke-20 nilai check tidak bernilai -1 maka percabangan bernilai false dan program tidak overflow. Kemudian program menampilkan pesan "Antrian berhasil ditambahkan" dan pesan "Queue

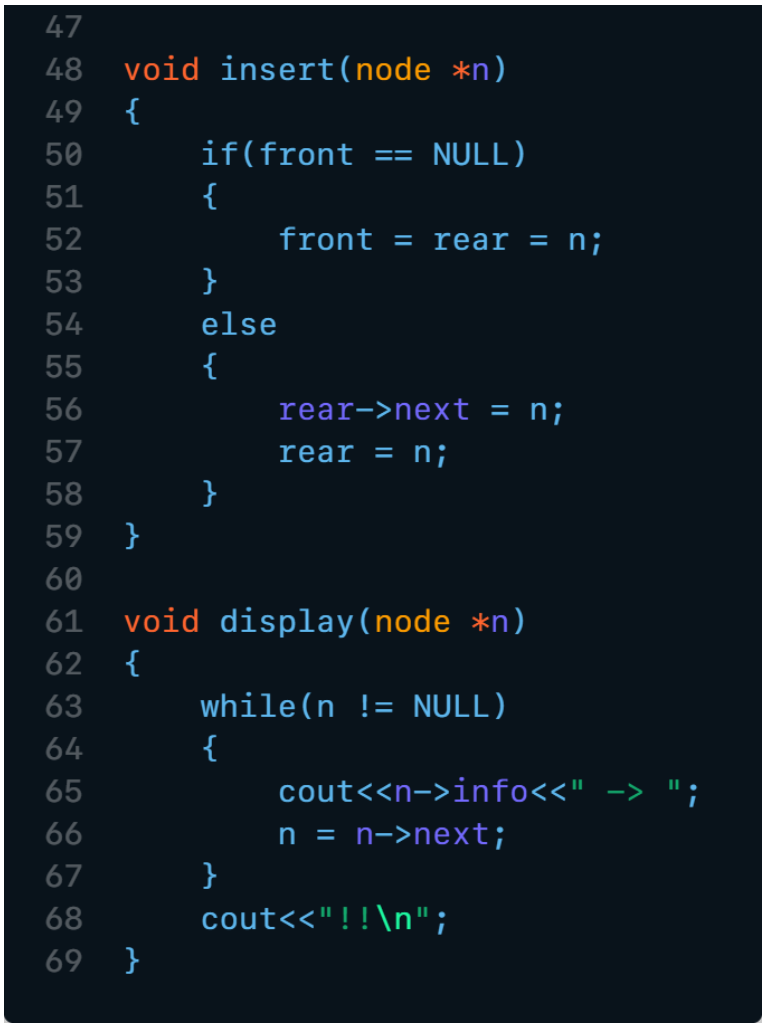
(depan…ke…belakang):". Kkemudian program menjalankan fungsi display untuk menampilkan antrian yang ada saat fungsi tersebut dipanggil. Kemudian program menampilkan pesan "Mau menambahkan antrian? (y/n)" dan meminta input user. Input user ini nantinya akan dijadikan acuan bagi program untuk memilih apakah akan mengulangi memasukkan elemen atau mengakhiri program. Kemudian user memasukkan nilai y yang berarti akan kembali memasukkan elemen ke dalam antrian. Karena user masukan nilai y maka perulangan bernilai true dan mengulang kembali meminta user untuk memasukkan elemen kepada antrian. Berikutnya user memasukkan nilai 90. Kemudian karena antrian masih belum penuh maka program kembali menampilkan "Antrian berhasil ditambahkan" dan pesan "Queue (depan…ke…belakang):" serta menampilkan antrian yang ada dengan memanggil fungsi display. Kemudian program kembali menanyakan kepada user apakah mau menambahkan antrian dan meminta user untuk memasukkan input. Kemudian user kembali memasukkan nilai y yang berarti user ingin kembali memasukkan elemen pada antrian. Karena user masukan nilai y maka perulangan bernilai true dan mengulang kembali meminta user untuk memasukkan elemen kepada antrian. Kemudian user memasukkan nilai 34 dan karena antrian masih belum penuh maka percabangan pada baris ke-20 masih bernilai false dan program tidak termasuk overflow. Kemudian program kembali menampilkan pesan Antrian berhasil ditambahkan" dan pesan "Queue (depan…ke…belakang):" dan kembali menampilkan antrian yang tersedia saat fungsi display dipanggil. Kemudian program kembali menanyakan kepada user apakah ingin menambahkan elemen pada antrian kemudian user memasukkan nilai n yang berarti user tidak ingin lagi memasukkan elemen pada antrian. Karena user memasukkan nilai n maka kondisi perulangan untuk masukkan elemen yang ada pada baris ke-15 bernilai false dan perulangan untuk memasukkan elemen pada antrian tidak dijalankan kembali oleh program. Kemudian program menampilkan pesan "Memulai pengeluaran elemen...". Pada saat ini program masuk ke perulangan yang berfungsi untuk mengeluarkan elemen dari antrian yang ada pada baris ke 34. Kemudian program menampilkan pesan "Elemen yang dikeluarkan dari antrian: " dan menampilkan nilai dari variabel check, yaitu 12. Nilai tersebut didapat dari menjalankan fungsi dequeue dengan argumen queue yang dimasukkan ke dalam variabel check. Karena

nilai dari variabel check bukan -1, maka kondisi percabangan pada baris ke 37 bernilai false sehingga program tidak menampilkan pesan underflow. Kemudian program menampilkan pesan “Queue (Depan...ke...Belakang):” dan menampilkan antrian yang ada pada saat itu dengan memanggil fungsi display dengan argument queue, front, dan rear. Kemudian program kembali menanyakan kepada user apakah ingin kembali mengeluarkan elemen dari antrian dan meminta input dari user. Input ini nantinya akan dijadikan acuan bagi program untuk memutuskan apakah ingin mengulang perulangan untuk mengeluarkan elemen atau mengakhiri program. Kemudian user memasukkan nilai y yang berarti user ingin kembali mengeluarkan elemen dari antrian. Kemudian program kembali menampilkan pesan “Elemen yang dikeluarkan dari antrian: “ dan menampilkan nilai dari variabel check, yaitu 90. Kemudian proGram kembali menampilkan pesan “Queue (Depan...ke...Belakang):” dan menampilkan antrian yang ada dengan memanggil fungsi display. Kemudian program kembali menanyakan kepada user apakah ingin kembali mengeluarkan elemen dari antrian dan meminta input dari user. Kemudian user kembali memasukkan nilai y yang berarti user ingin kembali mengeluarkan elemen dari antrian. Kemudian program kembali menampilkan pesan “Elemen yang dikeluarkan dari antrian: “ dan menampilkan nilai dari variabel check yang merupakan return dari pemanggilan fungsi dequeue, yaitu 34. Kemudian program menampilkan pesan “Queue (Depan...ke...Belakang):”. Kemudian program seharusnya menampilkan antrian yang ada saat ini dengan memanggil fungsi display. Namun karena seluruh data pada antrian sudah dikeluarkan maka fungsi display tidak mengembalikan nilai apapun. Kemudian program kembali menanyakan kepada user apakah ingin kembali mengeluarkan elemen dari antrian dan meminta input dari user. Kemudian user kembali memasukkan nilai y yang berarti user ingin kembali mengeluarkan elemen dari antrian. Kemudian karena nilai check yang didapatkan dari menjalankan fungsi dequeue adalah -1 maka kondisi pada percabangan baris ke 37 menjadi bernilai true dan program menampilkan pesan “Underflow..!!..Keluar program..” dan menjalankan exit(2). Kemudian program berakhir dan mengembalikan nilai 0 yang berarti program berjalan dengan normal.

5.4 Percobaan 6-4: Queue dengan Linked List Enqueue

5.4.a Source Code Percobaan 6-4: Queue dengan Linked List Enqueue

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

struct node
{
    int info;
    node *next;
} *front, *newptr, *save, *ptr, *rear;

node *create_new_node(int);
void insert(node *);
void display(node *);

int main()
{
    front = rear = NULL;
    int inf;
    int count=0;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan data pada antrian= ";
        cin>>inf;
        newptr = create_new_node(inf);
        if(newptr == NULL)
        {
            cout<<"Tidak dapat membuat node..!!..Keluar program.." << endl;
            exit(1);
        }
        insert(newptr);
        cout<<"Antrian berhasil ditambahkan..." << endl;
        cout<<"Queue (Depan...ke...Belakang):" << endl;
        display(front);
        cout<<"Mau menambahkan antrian? (y/n) ";
        cin>>ch;
    }
    return 0;
}

node *create_new_node(int x)
{
    ptr = new node;
    ptr->info = x;
    ptr->next = NULL;
    return ptr;
}
```



```
47
48  void insert(node *n)
49  {
50      if(front == NULL)
51      {
52          front = rear = n;
53      }
54      else
55      {
56          rear->next = n;
57          rear = n;
58      }
59  }
60
61  void display(node *n)
62  {
63      while(n != NULL)
64      {
65          cout<<n->info<<" -> ";
66          n = n->next;
67      }
68      cout<<"!!\n";
69  }
```

Gambar 5.4.a Source Code Queue dengan Linked List Enqueue

Berdasarkan gambar 5.4.a yang merupakan source code dari program queue dengan linked list enqueue, dapat dilihat pada baris ke 1 terdapat penggunaan library iostream yang berfungsi untuk input dan output dari user. Pada baris ke 2 terdapat penggunaan penamaan standar bagi compiler. Pada baris ke 4 terdapat inisialisasi struct node dengan atribut integer info dan node *next serta variabel *front, *newptr, *save, *ptr, dan *rear. Pada baris ke 10 terdapat inisialisasi node create_new_node dengan parameter variabel integer. Pada baris ke 11 terdapat deklarasi fungsi void insert dengan parameter node *. Pada baris ke 12 terdapat deklarasi fungsi void display dengan parameter node *. Pada baris ke 14 terdapat inisialisasi fungsi integer main. Pada baris ke 16 terdapat inisialisasi nilai variabel front = rear = NULL. Pada baris ke 17 terdapat deklarasi variabel integer inf. Pada baris ke 18 terdapat inisialisasi variabel count dengan nilai 0. Pada baris ke 19

terdapat inisialisasi variabel char ch dengan nilai y. Pada baris ke 20 terdapat perulangan while dengan kondisi nilai variabel ch = y atau Y. pada baris ke 22, di dalam perulangan terdapat perintah untuk menampilkan pesan “Masukkan data pada antrian= “ dan pada baris berikutnya terdapat perintah agar user untuk memasukkan nilai ke variabel inf. Nilai dari variabel ini nantinya akan menjadi nilai yang dimasukkan ke dalam antrian. Kemudian pada baris ke 24 terdapat inisialisasi nilai variabel newptr = create_new_node(inf). Pada baris ke 25 terdapat percabangan if dengan kondisi jika newptr = NULL maka tampilkan pesan “Tidak dapat membuat node..!!..Keluar program..” dan exit 1. Pada baris ke 30 terdapat perintah untuk menjalankan fungsi insert dengan argumen newptr. Pada baris ke 31 terdapat perintah untuk menampilkan pesan “Antrian berhasil ditambahkan...”. Pada baris ke 32 terdapat perintah untuk menampilkan pesan "Queue (Depan...ke...Belakang):" dan menjalankan fungsi display dengan argumen front. Pada baris ke 34 terdapat perintah untuk menampilkan pesan "Mau menambahkan antrian? (y/n) " dan pada baris ke 35 terdapat perintah agar user untuk memasukkan nilai ke variabel ch. Input user ini nantinya akan menentukan apakah program akan mengulangi perorangan untuk memasukkan data pada antrian atau mengakhiri program. Kemudian pada baris ke 37 terdapat return 0 yang berarti program berjalan dengan normal dan tidak ada kendala. Pada baris ke 40 terdapat inisialisasi node create_new_node yang berfungsi untuk membuat elemen data baru dengan parameter variabel integer x. Di dalamnya, pada baris ke 42 terdapat inisialisasi nilai variabel ptr = new node. Pada baris ke 43 terdapat inisialisasi nilai ptr->info = x. Pada baris ke 44 terdapat inisialisasi nilai ptr->next = NULL. Pada baris ke 45, fungsi mengembalikan nilai dari variabel ptr. Pada baris ke 48 terdapat inisialisasi fungsi void insert yang berfungsi untuk memasukkan elemen ke dalam antrian dengan parameter node *n. Pada baris ke 50, di dalam fungsi, terdapat percabangan if dengan kondisi jika front = NULL, maka inisialisasi nilai front = rear = n. Jika tidak, maka inisialisasi nilai rear->next = n dan rear = n. Pada baris ke 61 terdapat inisialisasi fungsi void display yang berfungsi untuk menampilkan antrian yang ada saat itu dengan parameter node *n. Pada baris ke 63, di dalam fungsi, terdapat perulangan while dengan kondisi selama n != NULL, maka tampilkan nilai dari n->info dan inisialisasi n = n->next.

5.4.b Output Percobaan 6-4: Queue dengan Linked List Enqueue

```
ueueLinkedListEnqueue } ; if ($?) { .\6-4queueLinkedListEnqueue }
Masukkan data pada antrian= 12
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12 -> !!
Mau menambahkan antrian? (y/n) y
Masukkan data pada antrian= 90
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12 -> 90 -> !!
Mau menambahkan antrian? (y/n) y
Masukkan data pada antrian= 34
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12 -> 90 -> 34 -> !!
Mau menambahkan antrian? (y/n) n
```

Gambar 5.4.b Output Queue dengan Linked List Enqueue

Berdasarkan gambar 5.4.b yang merupakan gambar dari output program queue dengan linked list enqueue, dapat dilihat bahwa pertama program menampilkan pesan “Masukkan data pada antrian” dan meminta user untuk memasukkan nilai. Kemudian user memasukkan nilai 12. Nilai ini nantinya akan dimasukkan ke dalam antrian. Kemudian program memanggil fungsi create_new_node dengan parameter input dari user dan menyimpannya ke dalam variabel newptr. Kemudian karena variabel newptr tidak bernilai NULL, maka kondisi percabangan pada baris ke 25 bernilai false dan program tidak menampilkan pesan “Tidak dapat membuat node”. Kemudian program menampilkan pesan “Antrian berhasil ditambahkan…” dan "Queue (Depan...ke...Belakang):", lalu menampilkan antrian yang ada dengan memanggil fungsi display dengan argument front. Kemudian program menampilkan pesan "Mau menambahkan antrian? (y/n) " dan meminta user untuk memasukkan nilai ke variabel ch. Nilai ini nantinya kana menjadi acuan bagi program untuk memutuskan apakah akan mengulangi perulangan utnuk memasukkan data atau menghentikan program. Kemudian user memasukkan nilai y yang berarti user ingin kembali memasukkan nilai ke dalam antrian. Kemudian program kembali meminta user untuk memasukkan data yang akan ditambahkan ke antrian. Kemudian user memasukkan nilai 90. Kemudian karena variabel newptr tidak bernilai NULL, maka kondisi percabangan pada baris ke 25 bernilai false dan program tidak menampilkan pesan “Tidak dapat membuat node”. Kemudian

program menampilkan pesan “Antrian berhasil ditambahkan…” dan "Queue (Depan...ke...Belakang):", lalu menampilkan antrian yang ada, yaitu 12 dan 90. Kemudian program menampilkan pesan "Mau menambahkan antrian? (y/n) " dan kembali meminta user untuk memasukkan nilai ke variabel ch. Kemudian user memasukkan nilai y yang berarti user ingin kembali memasukkan nilai ke dalam antrian. Kemudian program kembali meminta user untuk memasukkan data yang akan ditambahkan ke antrian. Kemudian user memasukkan nilai 34. Kemudian karena variabel newptr tidak bernilai NULL, maka kondisi percabangan pada baris ke 25 bernilai false dan program tidak menampilkan pesan “Tidak dapat membuat node”. Kemudian program menampilkan pesan “Antrian berhasil ditambahkan…” dan "Queue (Depan...ke...Belakang):", lalu menampilkan antrian yang ada, yaitu 12, 90, dan 34. Kemudian program menampilkan pesan "Mau menambahkan antrian? (y/n) " dan kembali meminta user untuk memasukkan nilai ke variabel ch. Kemudian user memasukkan nilai n yang berarti user tidak lagi ingin memasukkan data ke antrian. Karena variabel ch bernilai n, maka kondisi perulangan pada baris ke 20 menjadi bernilai false dan program tidak mengulang perulangan untuk memasukkan data lagi. Kemudian program berakhir.

5.5 Percobaan 6-5: Queue dengan Linked List Dequeue

5.5.a Source Code Percobaan 6-5: Queue dengan Linked List Dequeue

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

struct node
{
    int info;
    node *next;
} *front, *newptr, *save, *ptr, *rear;

node *create_new_node(int);
void insert(node *);
void delete_node_queue();
void display(node *);

int main()
{
    front = rear = NULL;
    int inf;
    int count=0;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout<<"Masukkan data pada antrian= ";
        cin>>inf;
        newptr = create_new_node(inf);
        if(newptr == NULL)
        {
            cout<<"Tidak dapat membuat node..!!..Keluar program.." << endl;
            exit(1);
        }
        insert(newptr);
        cout<<"Antrian berhasil ditambahkan..." << endl;
        cout<<"Queue (Depan...ke...Belakang):" << endl;
        display(front);
        cout<<"Mau menambahkan antrian? (y/n) ";
        cin>>ch;
    }

    do {
        cout<<"Queue (Depan...ke...Belakang):" << endl;
        display(front);
        if(count == 0)
        {
            cout<<"Mau mengeluarkan elemen dari antrian? (y/n) ";
            count++;
        }
        else
        {
            cout<<"Mau mengeluarkan elemen dari antrian lagi? (y/n) ";
        }
        cin>>ch;
        if(ch=='y' || ch=='Y')
        {
```

```
            delete_node_queue();
        }
        cout << endl;
    } while(ch=='y' || ch=='Y');
    return 0;
}

node *create_new_node(int x)
{
    ptr = new node;
    ptr->info = x;
    ptr->next = NULL;
    return ptr;
}

void insert(node *n)
{
    if(front == NULL)
    {
        front = rear = n;
    }
    else
    {
        rear->next = n;
        rear = n;
    }
}

void delete_node_queue()
{
    if(front == NULL)
    {
        cout<<"Overflow..!!..Keluar program.." << endl;
        exit(2);
    }
    else
    {
        ptr = front;
        front = front->next;
    }
}

void display(node *n)
{
    while(n != NULL)
    {
        cout<<n->info<<" -> ";
        n = n->next;
    }
    cout<<"!!\n";
}
```

Gambar 5.45.a Source Code Queue dengan Linked List Enqueue

Berdasarkan gambar 5.5.a yang merupakan source code dari program queue dengan linked list dequeue, dapat dilihat pada baris ke 1 terdapat penggunaan library iostream yang berfungsi untuk input dan output dari user. Pada baris ke 2 terdapat penggunaan penamaan standar bagi compiler. Pada baris ke 4 terdapat inisialisasi struct node dengan atribut integer info dan node *next serta variabel *front, *newptr, *save, *ptr, dan *rear. Pada baris ke 10 terdapat inisialisasi node create_new_node dengan parameter variabel integer. Pada baris ke 11 terdapat deklarasi fungsi void insert dengan parameter node *. Pada baris ke 12 terdapat deklarasi fungsi void delete_node_queue. Pada baris ke 13 terdapat deklarasi fungsi void display dengan parameter node *. Pada baris ke 15 terdapat inisialisasi fungsi integer main. Pada baris ke 17 terdapat inisialisasi nilai variabel front = rear = NULL. Pada baris ke 18 terdapat deklarasi variabel integer inf. Pada baris ke 19 terdapat inisialisasi variabel count dengan nilai 0. Pada baris ke 20 terdapat inisialisasi variabel char ch dengan nilai y. Pada baris ke 21 terdapat perulangan while dengan kondisi nilai variabel ch = y atau Y. pada baris ke 23, di dalam perulangan terdapat perintah untuk menampilkan pesan “Masukkan data pada antrian= “ dan pada baris berikutnya terdapat perintah agar user untuk memasukkan nilai ke variabel inf. Nilai dari variabel ini nantinya akan menjadi nilai yang dimasukkan ke dalam antrian. Kemudian pada baris ke 25 terdapat inisialisasi nilai variabel newptr = create_new_node(inf). Pada baris ke 26 terdapat percabangan if dengan kondisi jika newptr = NULL maka tampilkan pesan “Tidak dapat membuat node..!!..Keluar program..” dan exit 1. Pada baris ke 31 terdapat perintah untuk menjalankan fungsi insert dengan argumen newptr. Pada baris ke 32 terdapat perintah untuk menampilkan pesan “Antrian berhasil ditambahkan...”. Pada baris ke 33 terdapat perintah untuk menampilkan pesan "Queue (Depan...ke...Belakang):" dan menjalankan fungsi display dengan argumen front. Pada baris ke 35 terdapat perintah untuk menampilkan pesan "Mau menambahkan antrian? (y/n) " dan pada baris ke 36 terdapat perintah agar user untuk memasukkan nilai ke variabel ch. Input user ini nantinya akan menentukan apakah program akan mengulangi perorangan untuk memasukkan data pada antrian atau mengakhiri program. Kemudian pada baris ke 39 terdapat perulangan do-while. Pada baris ke 40, di dalam perulangan, terdapat perintah untuk menampilkan pesan "Queue

(Depan...ke...Belakang):". Pada baris ke 41 terdapat perintah untuk memanggil fungsi display dengan argument front. Pada baris ke 42 terdapat percabangan if dengan kondisi jika nilai count = 0 maka tampilkan pesan “Mau mengeluarkan elemen dari antrian? (y/n)” dan melakukan increment terhadap nilai count. Kemudia jika tidak maka program akan menampilkan pesan “Mau mengeluarkan elemen dari antrian lagi? (y/n)”. Pada baris ke 51 terdapat perintah agar user memasukkan nilai ke variabel ch. Pada baris ke 52 terdapat pervabangan if dengan kondisi jika variabel ch bernilai y atau Y maka jalankan fungsi delete_node_queue. Pada baris ke 57 terdapat kondisi dari perulangan do while, yaitu selama variabel ch bernilai y atau Y. Kemudian pada baris ke 58 terdapat return 0 yang berarti program berjalan dengan normal dan tidak ada kendala. Pada baris ke 61 terdapat inisialisasi node create_new_node dengan parameter variabel integer x. Di dalamnya, pada baris ke 63 terdapat inisialisasi nilai variabel ptr = new node. Pada baris ke 64 terdapat inisialisasi nilai ptr->info = x. Pada baris ke 65 terdapat inisialisasi nilai ptr->next = NULL. Pada baris ke 66, fungsi mengembalikan nilai dari variabel ptr. Pada baris ke 69 terdapat inisialisasi fungsi void insert dengan parameter node *n. Pada baris ke 72, di dalam fungsi, terdapat percabangan if dengan kondisi jika front = NULL, maka inisialisasi nilai front = rear = n. Jika tidak, maka inisialisasi nilai rear->next = n dan rear = n. Pada baris ke 82 terdapat inisialisasi fungsi void delete_node_queue. Pada baris ke 84, terdapat percabangan if dengan kondisi jika front = NULL maka tampilkan pesan “Overflow..!!..Keluar program.." dan jalankan exit(2). Jika tidak maka inisialisasi nilai ptr = front dan front = front->next. Pada baris ke 96 terdapat inisialisasi fungsi void display dengan parameter node *n. Pada baris ke 98, di dalam fungsi, terdapat perulangan while dengan kondisi selama n != NULL, maka tampilkan nilai dari n->info dan inisialisasi n = n->next.

5.5.b Output Percobaan 6-5: Queue dengan Linked List Dequeue

```
ueueLinkedListDequeue } ; if ($?) { .\6-5queueLinkedListDequeue }
Masukkan data pada antrian= 12
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12 -> !!
Mau menambahkan antrian? (y/n) y
Masukkan data pada antrian= 90
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12 -> 90 -> !!
Mau menambahkan antrian? (y/n) y
Masukkan data pada antrian= 34
Antrian berhasil ditambahkan...
Queue (Depan...ke...Belakang):
12 -> 90 -> 34 -> !!
Mau menambahkan antrian? (y/n) n
Queue (Depan...ke...Belakang):
12 -> 90 -> 34 -> !!
Mau mengeluarkan elemen dari antrian? (y/n) y

Queue (Depan...ke...Belakang):
90 -> 34 -> !!
Mau mengeluarkan elemen dari antrian lagi? (y/n) y

Queue (Depan...ke...Belakang):
34 -> !!
Mau mengeluarkan elemen dari antrian lagi? (y/n) y

Queue (Depan...ke...Belakang):
!!
Mau mengeluarkan elemen dari antrian lagi? (y/n) y
Overflow..!!..Keluar program..
```

Gambar 5.5.b Output Queue dengan Linked List Dequeue

Berdasarkan gambar 5.5.b yang merupakan gambar dari output program queue dengan linked list dequeue, dapat dilihat bahwa pertama program menampilkan pesan “Masukkan data pada antrian” dan meminta user untuk memasukkan nilai. Kemudian user memasukkan nilai 12. Nilai ini nantinya akan dimasukkan ke dalam antrian. Kemudian program memanggil fungsi create_new_node dengan parameter input dari user dan menyimpannya ke dalam variabel newptr. Kemudian karena variabel newptr tidak bernilai NULL, maka kondisi percabangan pada baris ke 25 bernilai false dan program tidak menampilkan pesan “Tidak dapat membuat node”. Kemudian program menampilkan pesan “Antrian berhasil ditambahkan…” dan "Queue (Depan...ke...Belakang):", lalu menampilkan antrian yang ada dengan memanggil fungsi display dengan argument front. Kemudian program

menampilkan pesan "Mau menambahkan antrian? (y/n) " dan meminta user untuk memasukkan nilai ke variabel ch. Nilai ini nantinya kana menjadi acuan bagi program untuk memutuskan apakah akan mengulangi perulangan utnuk memasukkan data atau menghentikan program. Kemudian user memasukkan nilai y yang berarti user ingin kembali memasukkan nilai ke dalam antrian. Kemudian program kembali meminta user untuk memasukkan data yang akan ditambahkan ke antrian. Kemudian user memasukkan nilai 90. Kemudian karena variabel newptr tidak bernilai NULL, maka kondisi percabangan pada baris ke 25 bernilai false dan program tidak menampilkan pesan "Tidak dapat membuat node". Kemudian program menampilkan pesan "Antrian berhasil ditambahkan…" dan "Queue (Depan...ke...Belakang):", lalu menampilkan antrian yang ada, yaitu 12 dan 90. Kemudian program menampilkan pesan "Mau menambahkan antrian? (y/n) " dan kembali meminta user untuk memasukkan nilai ke variabel ch. Kemudian user memasukkan nilai y yang berarti user ingin kembali memasukkan nilai ke dalam antrian. Kemudian program kembali meminta user untuk memasukkan data yang akan ditambahkan ke antrian. Kemudian user memasukkan nilai 34. Kemudian karena variabel newptr tidak bernilai NULL, maka kondisi percabangan pada baris ke 25 bernilai false dan program tidak menampilkan pesan "Tidak dapat membuat node". Kemudian program menampilkan pesan "Antrian berhasil ditambahkan…" dan "Queue (Depan...ke...Belakang):", lalu menampilkan antrian yang ada, yaitu 90 dan 34. Kemudian program menampilkan pesan "Mau menambahkan antrian? (y/n) " dan kembali meminta user untuk memasukkan nilai ke variabel ch. Kemudian user memasukkan nilai n yang berarti user tidak lagi ingin memasukkan data ke antrian. Karena variabel ch bernilai n, maka kondisi perulangan pada baris ke 20 menjadi bernilai false dan program tidak mengulang perulangan untuk memasukkan data lagi. Kemudian program menampilkan pesan "Queue (Depan...ke...Belakang):", lalu menampilkan antrian yang ada, yaitu 12, 90, dan 34. Kemudian program menmpilkan pesan "Mau mengeluarkan elemen dari antrian? (y/n)" dan meminta input dari user untuk variabel ch. Kemudian user memasukkan nilai y yang berarti user ingin mengeluarkan elemen dari antrian. Kemudian karena user memasukkan nilai y, maka kondisi percabangan pada baris ke 52 menjadi true dan program menjalankan fungsi delete_node_queue. Pada fungsi ini, data yang

berada paling depan saat ini, yaitu 12, seolah-olah dihapuskan dari antrian. Kemudian karena user memasukkan nilai y, maka kondisi perulangan while pada baris ke 57 menjadi true dan program kembali mengulang perulangan untuk mengeluarkan elemen dari antrian. Kemudian program menampilkan pesan "Queue (Depan...ke...Belakang):", lalu menampilkan antrian yang ada, yaitu 90 dan 34. Kemudian program menmpilkan pesan “Mau mengeluarkan elemen dari antrian lagi? (y/n)” dan meminta input dari user untuk variabel ch. Kemudian user memasukkan nilai y yang berarti user ingin kembali mengeluarkan elemen dari antrian. Kemudian karena user memasukkan nilai y, maka kondisi percabangan pada baris ke 52 menjadi true dan program menjalankan fungsi delete_node_queue. Pada fungsi ini, data yang berada paling depan saat ini, yaitu 90, seolah-olah dihapuskan dari antrian. Kemudian karena user memasukkan nilai y, maka kondisi perulangan while pada baris ke 57 menjadi true dan program kembali mengulang perulangan untuk mengeluarkan elemen dari antrian. Kemudian program menampilkan pesan "Queue (Depan...ke...Belakang):", lalu menampilkan antrian yang ada, yaitu 34. Kemudian program menmpilkan pesan “Mau mengeluarkan elemen dari antrian lagi? (y/n)” dan meminta input dari user untuk variabel ch. Kemudian user memasukkan nilai y yang berarti user ingin kembali mengeluarkan elemen dari antrian. Kemudian karena user memasukkan nilai y, maka kondisi percabangan pada baris ke 52 menjadi true dan program menjalankan fungsi delete_node_queue. Pada fungsi ini, data yang berada paling depan saat ini, yaitu 34, seolah-olah dihapuskan dari antrian. Kemudian karena user memasukkan nilai y, maka kondisi perulangan while pada baris ke 57 menjadi true dan program kembali mengulang perulangan untuk mengeluarkan elemen dari antrian. Kemudian program menampilkan pesan "Queue (Depan...ke...Belakang):", lalu menampilkan antrian yang ada. Karena data pada antrian telah kosong, program hanya menampilkan “!!”. Kemudian program kembali menanyakan ke user apakah ingin kembali mengeluarkan elemen dari antrian dan meminta input dari user. Kemudian user memasukkan nilai y yang berarti user ingin kembali mengeluarkan elemen dari antrian. Kemudian karena user memasukkan nilai y, maka kondisi percabangan pada baris ke 52 menjadi true dan program menjalankan fungsi delete_node_queue. Pada fungsi ini, di dalam percabangan pada baris ke 84, karena

data yang berada di antrian paling depan sudah kosong (NULL), maka program menampilkan pesan "Overflow..!!..Keluar program.." dan menjalankan exit(2) sehingga program berakhir.

VI. KESIMPULAN

Adapun kesimpulan dari percobaan ini adalah sebagai berikut :

1. Berdasarkan percobaan 5.1 Queue dengan Struct, dapat disimpulkan bahwa pendeklarasian variabel global #define MAX 8 berguna untuk memberikan batas maksimal dari antrian program.
2. Berdasarkan percobaan 5.1 Queue dengan Struct, dapat disimpulkan bahwa fungsi isFull berguna untuk melakukan pengecekan apakah data dalam queue sudah penuh atau belum. Sedangkan fungsi isEmpty berguna untuk melakukan pengecekan apakah data dalam queue masih kosong atau tidak.
3. Berdasarkan percobaan 5.1 Queue dengan Struct, dapat disimpulkan bahwa fungsi clear digunakan untuk mengosongkan semua data pada antrian dengan cara memberikan nilai -1 pada head dan tail sehingga tidak ada posisi data pada antrian.
4. Berdasarkan percobaan 5.4 Queue dengan Linked List Enqueue, dapat disimpulkan bahwa pada queue dengan menggunakan linked list, node yang baru dibuat akan ditambahkan pada ekor list.
5. Berdasarka percobaan 5.2 Queue dengan Array Enqueue dan percobaan 5.3 Queue dengan Array Dequeue dapat disimpulkan bahwa penggunaan variabel “const int SIZE” berguna untuk mengatur jumlah elemen maksimum yang ada pada queue.

DAFTAR PUSTAKA

Erliansyah Nasution dan Indra Yatini B.2005. Algoritma dan Struktur Data. Graha Ilmu. Yogyakarta.

Sjukani, Moh. 2010. (Algoritma Dan Struktur Data I) Dengan C, C++ dan Java. Jakarta: Mitra Wacana Media

Muhammad, M.A. (018. Struktur Data. Modul Struktur Data Unila PSTI. 2 (2) : 1 - 2.

Muhammad, Meizano Ardhi. 2018. Modul Praktikum Struktur Data. Lampung Jurusan Teknik Elektro Fakultas Teknik Universitas Lampung.

Ulum, M. Bahrul Ulum.2018. Modul Praktikum Struktur Data. Jakarta Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Esa Unggul.

TUGAS AKHIR

1. Source code

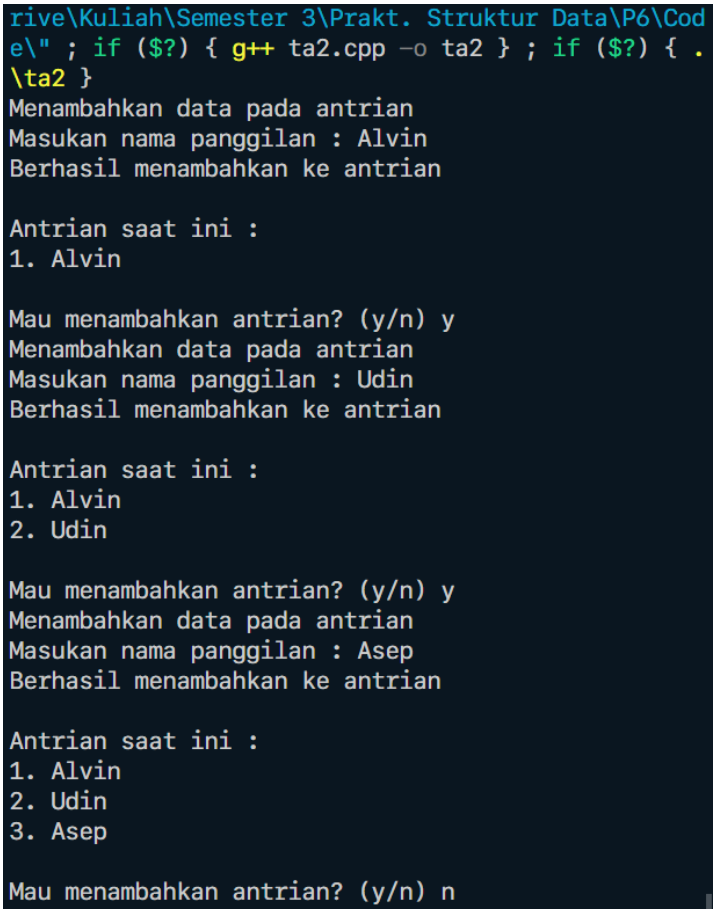
```cpp
#include<iostream>
using namespace std;

struct node
{
    int pos;
    string name;
    node *next;
} *front, *newptr, *save, *ptr, *rear;

node *create_new_node(string nama, int posisi);
void insert(node *);
void display(node *);

int main()
{
    front = rear = NULL;
    string name;
    int count = 1;
    char ch='y';
    while(ch=='y' || ch=='Y')
    {
        cout << "Menambahkan data pada antrian" << endl;
        cout<<"Masukan nama panggilan : ";
        cin>>name;
        newptr = create_new_node(name, count);
        if(newptr == NULL)
        {
            cout<<"Mohon maaf, tidak dapat menambahkan data
                pada antrian. Mengakhiri program..." << endl;
            exit(1);
        }
        insert(newptr);
        count++;
        cout<<"Berhasil menambahkan ke antrian" << endl << endl;
        display(front);
        cout<<"Mau menambahkan antrian? (y/n) ";
        cin>>ch;
    }
    return 0;
}

node *create_new_node(string name, int posisi)
{
    ptr = new node;
    ptr->name = name;
    ptr->pos = posisi;
    ptr->next = NULL;
    return ptr;
}

void insert(node *n)
{
    if(front == NULL)
    {
        front = rear = n;
    }
    else
    {
        rear->next = n;
        rear = n;
    }
}

void display(node *n)
{
    cout << "Antrian saat ini :" << endl;
    while(n != NULL)
    {
        cout <<n->pos << ". " << n->name << endl;
        n = n->next;
    }
    cout << endl;
}
```

Gambar 1 Source Code Program Antrian Pasien

Berdasarkan gambar 1 yang merupakan source code program antrian pasien, dapat dilihat bahwa pada baris ke 1 terdapat library iostream yang berguna untuk input dan output program. Pada baris ke 2 terdapat perintah untuk menggunakan penamaan standar bagi compiler. Pada baris ke 4 terdapat inisialisasi struct node dengan atribut integer pos sebagai posisi nomor arntrian, string name sebagai nama dari pasien, dan node *next sebagai address dari node berikutnya. Selain itu juga terdapat variabel front, newptr, save, ptr dan rear. Pada baris ke 11 terdapat deklarasi node create_new_node yang berfungsi untuk membuat elemen data baru dengan parameter string nama dan integer posisi. Pada baris ke 12 terdapat deklarasi fungsi void insert yang berfungsi untuk memasukkan elemen data yang telah dibuat ke antrian dengan parameter node *. Pada baris ke 13 terdapat deklarasi fungsi void display dengan parameter node *. Pada baris ke 15 terdapat insialisai fungsi integer main. Pada baris ke 17 terdapat inisialsisasi nilai front = rear = NULL. Pada baris ke 18 terdapat deklarasi variabel string name. Pada baris ke 19 terdapat inisialisasi variabel integer count dengan nilai 1. Pada baris ke 20 terdapat inisialisasi variabel char ch dengan nilai y. Pada baris ke 21 terdapat perulangan while dengan kondisi nilai ch = y atau Y. Pada baris ke 23, di dalam perulangan, terdapat perintah untuk menampilkan pesan “Menambahkan data pada antrian”. Pada baris ke 24 terdapat perintah untuk menampilkan pesan “Masukkan nama panggilan : “. Pada baris ke 25 tedapet perintah bagi user agar memasukkan nilai ke variabel name. Pada baris ke 26 terdapat inisialiasi nilai variabel newptr dengan hasil dari menajalnkan fungsi create_new_node dengan argumen name dan count. Pada baris ke 27 terdapat percabangan if dengan kondisi jika newptr = NULL maka tampilkan pesan "Mohon maaf, tidak dapat menambahkan data pada antrian. Mengakhiri program..." dan perintah exit(1). Pada baris ke 33 terdapat perintah untuk menjalankan fungsi insert dengan argumen newptr. Pada baris ke 34 terdapat increment pada nilai variabel count. Pada baris ke 35 terdapat perintah untuk menampilkan pesan “Berhasil menambahkan ke antrian”. Pada baris ke 36 terdapat perintah untuk menjalankan fungsi display dengan argumen front. Pada baris ke 37 terdapat perintah untuk menampilkan pesan “Mau menambahkan elemen? (y/n)”. Pada baris ke 38 terdapat perintah untuk agar user memasukkan nilai ke variabel ch. Pada baris ke 40 terdapat return 0 yang berarti program berakhir dengan normal. Pada baris ke 43 terdapat

inisialisasi node create_new_node dengan parameter string name dan integer posisi. Pada bari ke 45, di dalam fungsi create_new_node, terdapat inisialisasi nilai ptr = new node. Pada baris ke 46 terdapat inisialisasi ptr->name = name. Pada baris ke 47 terdapat inisialiasi nilai ptr->pos = posisi. Pada baris ke 48 terdapat inisialiasi nilai ptr->next = NULL. Kemudian berdasarkan baris ke 49, fungsi mengembalikan nilai ptr. Pada baris ke 52 terdapat inisialisasi fungsi void insert dengan paramater node *n. Pada baris ke 54 terdapat percabangan if dengan kondisi jika front = NULL maka inisialiasi nilai front = rear = n. Jika tidak maka inisialisasi nilai rear->next = n dan rear = n. Pada baris ke 65 terdapat inisialiasi fungsi void display degan parameter node *n. Pada baris ke 67, di dalam fungsi tersebut, terdapat perintah untuk menampilkan pesan "Antrian saat ini :" dan pada baris ke 68 terdapat perulangan while dengan kondisi selama nilai n != NULL, maka tampilkan nilai dari n->pos dan nilai dari n->name serta inisialisasikan nilai n = n->next.

2. Output



Gambar 2 Output Program Antrian Pasien

Berdasarkan gambar 2 yang merupakan output dari program antrian pasien, dapat dilihat bahwa pertama program menampilkan pesan “Menambahkan data pada antrian” yang berarti saat ini program akan memasukkan data yang diinputkan ke antrian. Kemudian program menampilkan pesan “Masukkan nama panggilan : “ dan meminta input dari user. Input dari user ini nantinya akan dimasukkan ke variabel name dan akan menjadi data yang akan dimasukkan ke dalam antrian. Kemudian user memasukkan nilai Alvin dan program memasukkan nilai tersebut sebagai argument dalam memanggil fungsi create_new_node. Kemudian program membuat elemen baru berdasarkan data yang diinputkan dan menyimpannya ke dalam variabel newptr. Kemudian karena nilai dari newptr tidak kosong, maka percabangan pada baris ke 27 tidak dijalankan dan program melanjutkan untuk memasukkan nilai ke dalam antrian dengan menajalankan fungsi insert dengan

newptr sebagai argumennya. Kemudian program melakukan increment kepada nilai count menjadi 2 yang berfungsi untuk menandakan bahwa program telah masuk ke antrian berikutnya ketika user akan memasukkan data kembali. Kemudian program menampilkan antrian yang ada dengan memanggil fungsi display dengan front sebagai argument. Kemudian program menampilkan pesan "Mau menambahkan antrian? (y/n)" dan meminta input dari user untuk dimasukkan ke variabel ch. Nilai yang diinputkan user akan menjadi acuan bagi program apakah akan mengulangi perulangan untuk menambahkan elemen atau tidak. Kemudian user memasukkan nilai y yang berarti user ingin kembali memasukkan data ke dalam antrian. Kemudian program kembali menampilkan pesan "Menambahkan data pada antrian" yang berarti saat ini program akan memasukkan data yang diinputkan ke antrian. Kemudian program menampilkan pesan "Masukkan nama panggilan : " dan meminta input dari user. Kemudian user memasukkan nilai Udin dan program memasukkan nilai tersebut sebagai argument dalam memanggil fungsi create_new_node. Kemudian program membuat elemen baru berdasarkan data yang diinputkan dan menyimpannya ke dalam variabel newptr. Kemudian karena nilai dari newptr tidak kosong, maka percabangan pada baris ke 27 tidak dijalankan dan program melanjutkan untuk memasukkan nilai ke dalam antrian dengan menajalankan fungsi insert dengan newptr sebagai argumennya. Kemudian program melakukan increment kepada nilai count menjadi 3. Kemudian program menampilkan antrian yang ada dengan memanggil fungsi display dengan front sebagai argument. Kemudian program menampilkan pesan "Mau menambahkan antrian? (y/n)" dan meminta input dari user untuk dimasukkan ke variabel ch. Kemudian user memasukkan nilai y yang berarti user ingin kembali memasukkan data ke dalam antrian. Kemudian program kembali menampilkan pesan menampilkan pesan "Menambahkan data pada antrian" yang berarti saat ini program akan memasukkan data yang diinputkan ke antrian. Kemudian program menampilkan pesan "Masukkan nama panggilan : " dan meminta input dari user. Kemudian user memasukkan nilai Asep dan program memasukkan nilai tersebut sebagai argument dalam memanggil fungsi create_new_node. Kemudian program membuat elemen baru berdasarkan data yang diinputkan dan menyimpannya ke dalam variabel newptr. Kemudian karena nilai dari newptr tidak kosong, maka

percabangan pada baris ke 27 tidak dijalankan dan program melanjutkan untuk memasukkan nilai ke dalam antrian dengan menajalankan fungsi insert dengan newptr sebagai argumennya. Kemudian program melakukan increment kepada nilai count menjadi 3. Kemudian program menampilkan antrian yang ada dengan memanggil fungsi display dengan front sebagai argument. Kemudian program menampilkan pesan “Mau menambahkan antrian? (y/n)” dan meminta input dari user untuk dimasukkan ke variabel ch. Kemudian user memasukkan nilai n yang berarti user tidak lagi ingin memasukkan data ke dalam antrian. Karena user memasukkan nilai n maka perulangan untuk menambahkan node pada baris ke 21 menjadi bernilai false dan program keluar dari perulangan tersebut. Kemudian program keluar dari return 0.