LAPORAN PARKTIKUM
MODUL VII ALGORITMA DAN STRUKTUR DATA
”QUEUE”

Disusun oleh :

Yasvin Syahgana (2311102065)

Dosen :

Wahyu Andi Syahputra, S.pd.,M.Eng

PROGRAM STUDI

S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA INSTITUT TEKNOLOGI

TELKOM PURWOKERTO 2024

BAB I
TUJUAN PARKTIKUM

Tujuan parktikum adalah sebagai berikut :

1. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
2. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
3. Mahasiswa mampu menerapkan operasi tampil data pada queue

BAB II

DASAR TEORI

A. Pengertian Queue

Queue adalah struktur data linear yang mengikuti prinsip First-In-First-Out (FIFO), yang berarti item pertama yang ditambahkan ke dalam queue akan menjadi item pertama yang dihapus.

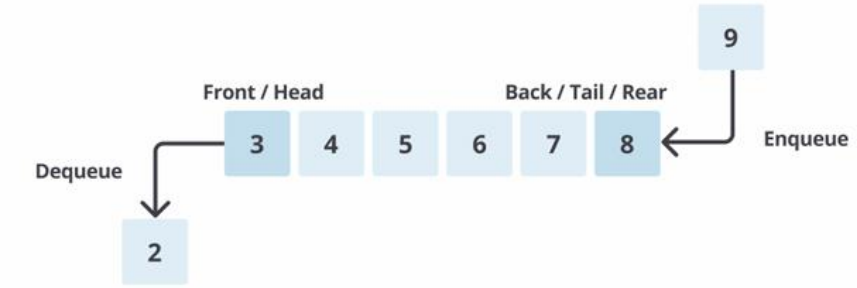
Queue digunakan untuk mengelola suatu urutan item dalam cara yang memungkinkan item ditambahkan dan dihapus dari ujung urutan.

B. Operasi Dasar

1. Enqueue : Menambahkan item ke akhir queue.
2. Dequeue : Menghapus item dari depan queue.
3. Front : Mengembalikan item di depan queue tanpa menghapusnya.
4. Rear : Mengembalikan item di ujung queue tanpa menghapusnya.
5. IsEmpty : Mengecek apakah queue kosong.

C. Implementasi

1. Implementasi Berbasis Array : Membuat queue menggunakan array, di mana pointer depan dan belakang digunakan untuk memantau keadaan queue.
2. Implementasi Berbasis Linked List : Membuat queue menggunakan linked list, di mana setiap node mewakili item dalam queue.



D. Kompleksitas Waktu

1. Enqueue : O(1) untuk implementasi berbasis array, O(1) untuk implementasi berbasis linked list.
2. Dequeue : O(1) untuk implementasi berbasis array, O(1) untuk

implementasi berbasis linked list.

3. Front : O(1) untuk implementasi berbasis array, O(1) untuk implementasi berbasis linked list.
4. Rear : O(1) untuk implementasi berbasis array, O(1) untuk implementasi berbasis linked list.
5. IsEmpty : O(1) untuk implementasi berbasis array, O(1) untuk E berbasis linked list.

E. Aplikasi

1. Jadwal Kerja : Queue digunakan dalam algoritma jadwal kerja untuk mengelola suatu urutan tugas yang perlu dijalankan dalam urutan tertentu.
2. Queue Cetak : Queue digunakan dalam queue cetak untuk mengelola suatu urutan dokumen yang perlu dicetak.
3. Protokol Jaringan : Queue digunakan dalam protokol jaringan untuk mengelola suatu urutan paket yang perlu diirimkan melalui jaringan.

F. Kelebihan

1. Efisiensi : Queue efisien dalam mengelola suatu urutan item karena memungkinkan penambahan dan penghapusan item yang cepat.
2. Flexibel : Queue fleksibel karena dapat diimplementasikan menggunakan struktur data yang berbeda seperti array atau linked list.

G. Kekurangan

Kapasitas Terbatas : Queue memiliki kapasitas terbatas, yang berarti hanya dapat menampung sejumlah item sebelum perlu diperluas atau dibersihkan.

Dequeue Lambat : Operasi dequeue dapat lambat jika queue besar dan item di depan queue perlu dipindahkan ke ujung queue.

BAB III

GUIDED & UNGUIDED

Guided

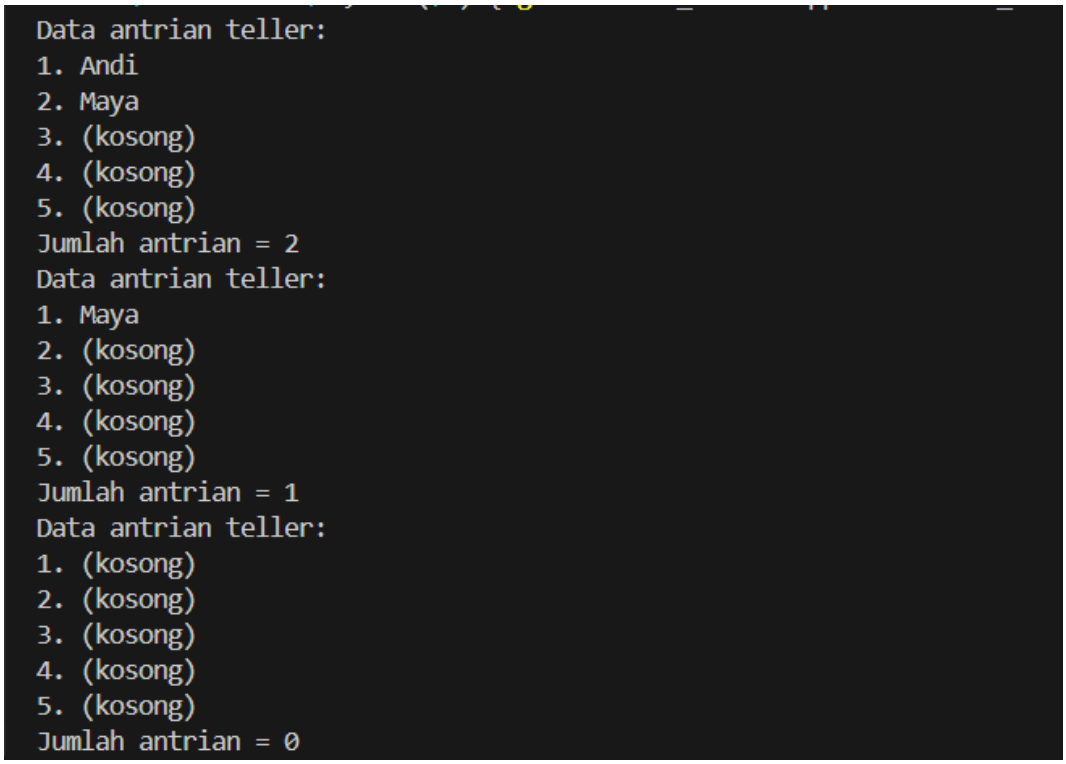
Source Code

```
// YASVIN SYAHGANA
// 2311102065
// GUIDED 1
#include <iostream>
using namespace std;

const int maksimalQueue = 5; // Maksimalantrian
int front_065 = 0;           // Penandaantrian
int back_065 = 0;            // Penanda
string queueTeller[5];       // Fungsipengecekan
bool isFull() {              //
Pengecekanantrianpenuhatautidak
  if (back_065 == maksimalQueue) {
    return true; // =1
  } else {
    return false;
  }
}
bool isEmpty() { // Antriannya kosong atau tidak
  if (back_065 == 0) {
    return true;
  } else {
    return false;
  }
}
void enqueueAntrian(string data) { // Fungsi menambahkan
antrian
  if (isFull()) {
    cout << "Antrian penuh" << endl;
  } else {
    if (isEmpty()) { // Kondisi ketika queue kosong
      queueTeller[0] = data;
      front_065++;
      back_065++;
    } else { // Antrianya ada isi
      queueTeller[back_065] = data;
      back_065++;
    }
  }
}
```

```
void dequeueAntrian() { // Fungsi mengurangi antrian
  if (isEmpty()) {
    cout << "Antrian kosong" << endl;
  } else {
    for (int i = 0; i < back_065; i++) {
      queueTeller[i] = queueTeller[i + 1];
    }
    back_065--;
  }
}
int countQueue() { // Fungsi menghitung banyak antrian
  return back_065;
}
void clearQueue() { // Fungsi menghapus semua antrian
  if (isEmpty()) {
    cout << "Antrian kosong" << endl;
  } else {
    for (int i = 0; i < back_065; i++) {
      queueTeller[i] = "";
    }
    back_065 = 0;
    front_065 = 0;
  }
}
void viewQueue() { // Fungsi melihat antrian
  cout << "Data antrian teller:" << endl;
  for (int i = 0; i < maksimalQueue; i++) {
    if (queueTeller[i] != "") {
      cout << i + 1 << ". " << queueTeller[i] << endl;
    } else {
      cout << i + 1 << ". (kosong)" << endl;
    }
  }
}
int main() {
  enqueueAntrian("Andi");
  enqueueAntrian("Maya");
  viewQueue();
  cout << "Jumlah antrian = " << countQueue() << endl;
  dequeueAntrian();
  viewQueue();
  cout << "Jumlah antrian = " << countQueue() << endl;
  clearQueue();
  viewQueue();
  cout << "Jumlah antrian = " << countQueue() << endl;
  return 0;
}
```

Output



Deskripsi Program

Program ini mengimplementasikan antrian (queue) menggunakan bahasa pemrograman C++. Berikut adalah deskripsi singkat dari setiap fungsinya:

Fungsi isFull():

- Mengecek apakah antrian penuh dengan membandingkan posisi belakang dengan kapasitas maksimum (maksimalQueue).

Fungsi isEmpty():

- Mengecek apakah antrian kosong dengan memeriksa apakah posisi belakang adalah 0.

Fungsi enqueueAntrian(string data):

- Menambahkan elemen ke antrian jika tidak penuh. Jika antrian kosong, elemen ditempatkan di posisi awal. Jika tidak, elemen ditempatkan di posisi belakang antrian.

Fungsi dequeueAntrian():

- Menghapus elemen dari antrian jika tidak kosong dengan menggeser semua elemen ke depan satu posisi.

Fungsi countQueue():

- Mengembalikan jumlah elemen dalam antrian dengan mengembalikan nilai back.

Fungsi clearQueue():

- Mengosongkan seluruh elemen antrian dengan mengatur semua posisi menjadi kosong dan mengatur penanda depan dan belakang ke 0.

Fungsi viewQueue():

- Menampilkan semua elemen dalam antrian, menandai posisi yang kosong.

Fungsi main():

- Menunjukkan penggunaan fungsi-fungsi di atas dengan menambahkan dua elemen ("Andi" dan "Maya"), menampilkan antrian, menghapus satu elemen, menampilkan antrian lagi, dan akhirnya mengosongkan antrian serta menampilkan antrian yang kosong.

UNGUIDED 1

Source Code

```
// YASVIN SYAHGANA
// 2311102065
// UNGUIDED 1

#include <iostream>

using namespace std;

struct Node {
  string data_065;
  Node *next_065;
};
class Queue {
private:
  Node *front_065;
  Node *rear_065;

public:
  Queue() {
    front_065 = NULL;
    rear_065 = NULL;
  }
  bool Empety_065() { return front_065 == NULL; }
  void enqueue(string data_065) {
    Node *newNode = new Node();
    newNode->data_065 = data_065;
    newNode->next_065 = NULL;

    if (Empety_065()) {
      front_065 = newNode;
      rear_065 = newNode;
    } else {
      rear_065->next_065 = newNode;
      rear_065 = newNode;
    }
    cout << " Mahasiswa " << data_065
         << "berhasil di tambahkan ke dalam antrian." <<
endl;
  }
  void dequeue_065() {
    if (Empety_065()) {
      cout << "antrian kosong" << endl;
    } else {
      Node *temp = front_065;
```

```
      cout << "Mahasiswa " << front_065->data_065
           << "telah selesai dilayani dan di keluarkan dari
antrian.\n"
           << endl;

      if (front_065 == rear_065) {
        front_065 = NULL;
        rear_065 = NULL;
      } else {
        front_065 = front_065->next_065;
      }
      delete temp;
    }
  }
  int countQueue() {
    int count = 0;
    Node *temp = front_065;
    while (temp != NULL) {
      count++;
      temp = temp->next_065;
    }
    return count;
  }
  void hapusQueue() {
    if (Empety_065()) {
      cout << "Antrian Kosong" << endl;
    } else {
      while (Empety_065()) {
        dequeue_065();
      }
      cout << "Seluruh Antrian telah di hapus." << endl;
    }
  }
  void tampilQueue() {
    if (Empety_065()) {
      cout << "Antrian kosong." << endl;
    } else {
      cout << "Data Antrian Mahasiswa:" << endl;
      Node *temp = front_065;
      int position = 1;
      while (temp != NULL) {
        cout << position << " Nama :" << temp->data_065 <<
endl;
        temp = temp->next_065;
        position++;
      }
    }
```

```
  }
};
int main() {
  Queue queue;
  queue.enqueue("Andi");
  queue.enqueue("Maya");
  queue.tampilQueue();
  cout << " Jumlah antrian : " << queue.countQueue() << endl;
  queue.dequeue_065();
  queue.tampilQueue();
  cout << " Jumlah antrian : " << queue.countQueue() << endl;
  queue.hapusQueue();
  queue.tampilQueue();
  cout << " Jumlah antrian : " << queue.countQueue() << endl;

  return 0;
}
```

Screenshoot Program

```
 Mahasiswa Andiberhasil di tambahkan ke dalam antrian.
  Mahasiswa Mayaberhasil di tambahkan ke dalam antrian.
 Data Antrian Mahasiswa:
 1 Nama :Andi
 2 Nama :Maya
  Jumlah antrian : 2
 Mahasiswa Anditelah selesai dilayani dan di keluarkan dari antrian.

 Data Antrian Mahasiswa:
 1 Nama :Maya
  Jumlah antrian : 1
 Seluruh Antrian telah di hapus.
 Data Antrian Mahasiswa:
 1 Nama :Maya
  Jumlah antrian : 1
```

Deskripsi Program

Program ini adalah implementasi Queue (Antrian) dalam bahasa C++ yang digunakan untuk mengelola urutan item (dalam hal ini, nama-nama orang) yang ditambahkan dan dihapus secara teratur. Queue ini memiliki kapasitas maksimal 5 item dan menggunakan array untuk menyimpan item.

Program ini memiliki fungsi-fungsi berikut:

1. enqueueAntrian(string data): Menambahkan item ke dalam queue.

Jika queue penuh, program akan menampilkan pesan "Antrian penuh".

2. dequeueAntrian(): Menghapus item dari depan queue. Jika queue kosong, program akan menampilkan pesan "Antrian kosong".
3. countQueue(): Menghitung jumlah item yang ada di dalam queue.
4. clearQueue(): Menghapus semua item di dalam queue.
5. viewQueue(): Menampilkan isi queue, termasuk item yang ada dan item yang kosong.

Dalam program ini, akan dapat melihat bagaimana queue digunakan untuk mengelola urutan item. Program ini juga menampilkan jumlah item yang ada di dalam queue sebelum dan setelah operasi enqueue dan dequeue dilakukan. Setelah operasi dequeue dilakukan, program juga menampilkan isi queue yang telah berubah. Akhirnya, program menghapus semua item di dalam queue menggunakan fungsi clearQueue() dan menampilkan isi queue yang kosong.

UNGUIDED 2

Source Code

```
// YASVIN SYAHGANA
// 2311102065
// UNGUIDED 2

#include <iostream>

using namespace std;

struct Node {
  string nama_065;
  string nim_065;
  Node *next_065;
};
Node *top;
void init() { top = NULL; }
bool isEmpety() { return top == NULL; }
void push(string nama_065, string nim_065) {
  Node *newNode = new Node;
  newNode->nama_065 = nama_065;
  newNode->nim_065 = nim_065;
  newNode->next_065 = top;
  top = newNode;
  cout << " Data Mahasiswa " << nama_065 << " dengan NIM "
<< nim_065
       << " berhasil  di tambahkan!." << endl;
}
void pop() {
  if (isEmpety()) {
    cout << "Data mahasiswa kosong!" << endl;
  } else {
    Node *temp = top;
    cout << "Data Mahasiswa " << temp->nama_065 << " dengan
NIM "
         << temp->nim_065 << "dihapus dari stack" << endl;
    top = top->next_065;
    delete temp;
  }
}
void reset() {
  while (isEmpety()) {
    pop();
  }
  cout << " Data mahasiswa berhasil di reset! " << endl;
}
```

```
void TAMPILKAN() {
  if (isEmpety()) {
    cout << " Data  Mahasiswa kosong! " << endl;
  } else {
    Node *current = top;
    int position = 1;
    while (current != NULL) {
      cout << position << " Nama :" << current->nama_065 <<
endl;
      cout << " NIM : " << current->nim_065 << endl;
      current = current->next_065;
      position++;
    }
  }
}
int main() {
  int choice;
  string nama_065, nim_065;
  init();
loop_menu:
  cout << "\n======= Data Mahasiswa =======" << endl;
  cout << " 1. Masukkan Data Mahasiswa " << endl;
  cout << " 2. Hapus Satu Data Mahasiwa " << endl;
  cout << " 3. Reset data Mahasiwa " << endl;
  cout << " 4. Tampilkan Data Mahasiswa " << endl;
  cout << " 5. Keluar. " << endl;
  cout << " Masukkan Pilihan Anda (1-5) = ";
  cin >> choice;
  cout << endl;
  switch (choice) {
  case 1:
    cout << " ======= Masukkan Data Mahasiswa =======" <<
endl;
    cout << " Masukkan Nama : ";
    cin >> nama_065;
    cout << " Masukkan NIM : ";
    cin >> nim_065;
    push(nama_065, nim_065);
    break;
  case 2:
    pop();
    break;
  case 3:
    reset();
    break;
  case 4:
    cout << "====Tampilkan Data Mahasiswa====" << endl;
```

```
    TAMPILKAN();
    break;
  case 5:
    cout << " Terimakasih!" << endl;

    return 0;
  default:
    cout << " Pilihan tidak valid " << endl;
  }
  goto loop_menu;
}
```

Screenshoot Program

1. Memasukkan Data

```
======= Data Mahasiswa =======
 1. Masukkan Data Mahasiswa
 2. Hapus Satu Data Mahasiwa
 3. Reset data Mahasiwa
 4. Tampilkan Data Mahasiswa
 5. Keluar.
 Masukkan Pilihan Anda (1-5) = 1

 ======= Masukkan Data Mahasiswa =======
 Masukkan Nama : Gana
 Masukkan NIM : 2311102065
 Data Mahasiswa Gana dengan NIM 2311102065 berhasil  di tambahkan!.

======= Data Mahasiswa =======
 1. Masukkan Data Mahasiswa
 2. Hapus Satu Data Mahasiwa
 3. Reset data Mahasiwa
 4. Tampilkan Data Mahasiswa
 5. Keluar.
 Masukkan Pilihan Anda (1-5) = 4

====Tampilkan Data Mahasiswa====
1 Nama :Gana
 NIM : 2311102065
```

```
======= Data Mahasiswa =======
 1. Masukkan Data Mahasiswa
 2. Hapus Satu Data Mahasiwa
 3. Reset data Mahasiwa
 4. Tampilkan Data Mahasiswa
 5. Keluar.
 Masukkan Pilihan Anda (1-5) = 1

 ======= Masukkan Data Mahasiswa =======
 Masukkan Nama : Adam
 Masukkan NIM : 2311102099
 Data Mahasiswa Adam dengan NIM 2311102099 berhasil  di tambahkan!.
```

2. Menghapus Satu data Mahasiswa Dari Stack

```
======= Data Mahasiswa =======
 1. Masukkan Data Mahasiswa
 2. Hapus Satu Data Mahasiwa
 3. Reset data Mahasiwa
 4. Tampilkan Data Mahasiswa
 5. Keluar.
 Masukkan Pilihan Anda (1-5) = 2

Data Mahasiswa Adam dengan NIM 2311102099dihapus dari stack
```

3. **Menampilkan Data Mahasiswa**

```
======= Data Mahasiswa =======
 1. Masukkan Data Mahasiswa
 2. Hapus Satu Data Mahasiwa
 3. Reset data Mahasiwa
 4. Tampilkan Data Mahasiswa
 5. Keluar.
 Masukkan Pilihan Anda (1-5) = 4

====Tampilkan Data Mahasiswa====
1 Nama :Gana
 NIM : 2311102065
```

BAB IV

Kesimpulan

Queue adalah struktur data linear yang mengikuti prinsip FIFO (First In, First Out). Dalam C++, implementasi dasar queue melibatkan operasi seperti:

1. Enqueue: Menambahkan elemen ke akhir antrian.
2. Dequeue: Menghapus elemen dari awal antrian.
3. IsFull: Mengecek apakah antrian penuh.
4. IsEmpty: Mengecek apakah antrian kosong.
5. Count: Menghitung jumlah elemen dalam antrian.
6. Clear: Mengosongkan antrian.
7. View: Menampilkan elemen dalam antrian.

Manfaat:

1. Mempertahankan urutan pemrosesan elemen.
2. Berguna dalam berbagai aplikasi seperti manajemen tugas, sistem antrian pelanggan, dan pemrosesan printer.
3. Keterbatasan:Antrian statis memiliki kapasitas tetap dan memerlukan penggeseran elemen saat dequeue.

Optimasi

Menggunakan linked list atau circular queue untuk mengatasi keterbatasan dan meningkatkan efisiensi. Queue adalah alat penting dalam pemrograman untuk mengelola data yang memerlukan urutan pemrosesan yang teratur dan efisien.

BAB V

DAFTAR PUSTAKA

1. https://en.cppreference.com/w/cpp/container/queue
2. https://www.geeksforgeeks.org/queue-data-structure/