LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VIII
QUEUE

Disusun Oleh :
NAMA : Afief Amar Purnomo
NIM : 103112430067

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Queue adalah struktur data linier yang mengikuti prinsip FIFO (First In First Out), artinya data yang pertama kali masuk akan menjadi data pertama yang keluar. Dalam queue, elemen ditambahkan di bagian belakang (tail) dan dihapus dari bagian depan (head). Struktur ini berbeda dengan stack yang menggunakan prinsip LIFO (Last In First Out).

Queue sangat berguna untuk mengelola data secara berurutan seperti antrean dalam dunia nyata dan digunakan di berbagai aplikasi sistem komputer, seperti penjadwalan tugas dan pengelolaan antrian layanan. Implementasi queue dapat dilakukan menggunakan array atau linked list, dengan variasi seperti circular queue untuk efisiensi pemakaian memori.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

(file queue.h)

queue.h > ...

```c
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5

struct Queue {
    int info[MAX_QUEUE];
    int head;
    int tail;
    int count;
};

Tabnine | Edit | Test | Explain | Document
void createQueue(Queue &Q);

Tabnine | Edit | Test | Explain | Document
bool isEmpty(Queue Q);

Tabnine | Edit | Test | Explain | Document
bool isFull(Queue Q);

Tabnine | Edit | Test | Explain | Document
void enqueue(Queue &Q, int x);

Tabnine | Edit | Test | Explain | Document
int dequeue(Queue &Q);

Tabnine | Edit | Test | Explain | Document
void printInfo(Queue Q);

#endif
```

(file queue.cpp)

queue.cpp > printInfo(Queue)

```cpp
#include "queue.h"
#include <iostream>

using namespace std;

Tabnine | Edit | Test | Explain | Document
void createQueue(Queue &Q) {
    Q.head = 0;
    Q.tail = 0;
    Q.count = 0;
}

Tabnine | Edit | Test | Explain | Document
bool isEmpty(Queue Q) {
    return Q.count == 0;
}

Tabnine | Edit | Test | Explain | Document
bool isFull(Queue Q) {
    return Q.count == MAX_QUEUE;
}

Tabnine | Edit | Test | Explain | Document
void enqueue(Queue &Q, int x) {
    if (!isFull(Q)) {
        Q.info[Q.tail] = x;
        Q.tail = (Q.tail + 1) % MAX_QUEUE;
        Q.count++;
    } else {
        cout << "Antrian Penuh" << endl;
    }
}
```

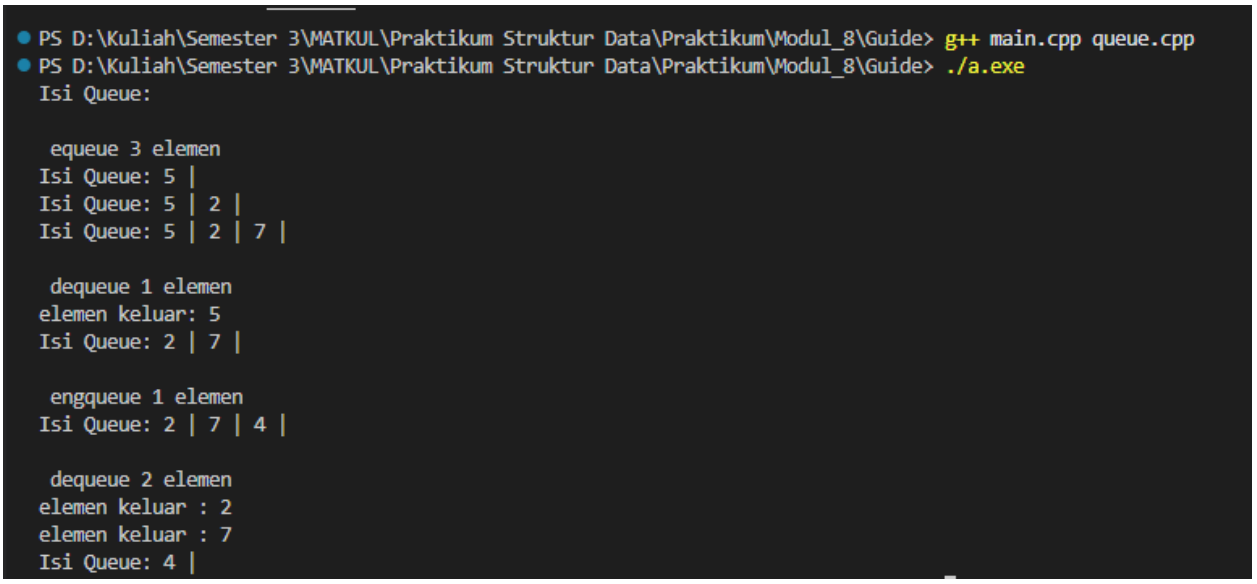
```
29
    Tabnine | Edit | Test | Explain | Document
30  int dequeue(Queue &Q) {
31      if (!isEmpty(Q)) {
32          int x = Q.info[Q.head];
33          Q.head = (Q.head + 1) % MAX_QUEUE;
34          Q.count--;
35          return x;
36      } else {
37          cout << "Antrian Kosong" << endl;
38          return -1;
39      }
40  }
41
    Tabnine | Edit | Test | Explain | Document
42  void printInfo(Queue Q) {
43      cout << "Isi Queue: ";
44      if (!isEmpty(Q)) {
45          int i = Q.head;
46          int n = 0;
47          while (n < Q.count) {
48              cout << Q.info[i] << " | ";
49              i = (i + 1) % MAX_QUEUE;
50              n++;
51          }
52      }
53      cout << endl;
54  }
```

(file main.cpp)

main.cpp > ...

```cpp
#include <iostream>
#include "queue.h"

using namespace std;

Tabnine | Edit | Test | Explain | Document
int main(){
    Queue Q;

    createQueue(Q);
    printInfo(Q);

    cout << "\n equeue 3 elemen" << endl;
    enqueue (Q, 5);
    printInfo(Q);
    enqueue (Q, 2);
    printInfo(Q);
    enqueue (Q, 7);
    printInfo(Q);

    cout << "\n dequeue 1 elemen" << endl;

    cout << "elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q);

    cout << "\n engqueue 1 elemen" << endl;
    enqueue(Q, 4);
    printInfo(Q);

    cout << "\n dequeue 2 elemen" << endl;
    cout << "elemen keluar : " << dequeue(Q) << endl;
    cout << "elemen keluar : " << dequeue(Q) << endl;
     printInfo(Q);

     return 0;

}
```

Screenshots Output



Deskripsi:

Program di atas merupakan implementasi struktur data queue (antrian) menggunakan bahasa C++ dengan teknik circular queue berbasis array. Struktur Queue memiliki tiga komponen utama: head sebagai indeks elemen depan, tail sebagai indeks elemen belakang, dan count untuk menghitung jumlah elemen yang sedang berada dalam antrian. Program menyediakan beberapa fungsi penting, yaitu enqueue untuk menambahkan elemen ke belakang antrian, dequeue untuk menghapus dan mengembalikan elemen dari depan antrian, isEmpty untuk memeriksa apakah antrian kosong, serta isFull untuk mengecek apakah antrian penuh. Selain itu terdapat fungsi printInfo untuk menampilkan seluruh isi antrian secara berurutan. Pada fungsi main, program membuat queue kosong, kemudian menambahkan elemen 5, 2, dan 7 secara berurutan menggunakan enqueue, sehingga outputnya menjadi “5 | 2 | 7 |”. Setelah dilakukan dequeue, elemen pertama yang keluar adalah 5 karena queue bersifat FIFO (First In, First Out). Selanjutnya dimasukkan elemen 4, dan setelah beberapa operasi dequeue berikutnya, isi antrian berubah sesuai urutan pemrosesan, menunjukkan cara kerja antrian yang memproses elemen berdasarkan giliran masuknya.

C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1 (soal 1)

(file queue.h)

queue.h > createQueue(Queue &)

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5
typedef int infotype;

struct Queue {
    infotype info[MAX_QUEUE];
    int head;
    int tail;
};

//Alternatif 1
void createQueue(Queue &Q);
bool isEmptyQueue(Queue Q);
bool isFullQueue(Queue Q);
void enqueue(Queue &Q, infotype x);
infotype dequeue(Queue &Q);

//Alternatif 2
void enqueue2(Queue &Q, infotype x);
infotype dequeue2(Queue &Q);

//Alternatif 3
void createQueue3(Queue &Q);
bool isFullQueue3(Queue Q);
void enqueue3(Queue &Q, infotype x);
infotype dequeue3(Queue &Q);

void printInfo(Queue Q);

#endif
```

(file queue.cpp)

queue.cpp > isEmptyQueue(Queue)

```cpp
#include <iostream>
#include <cstdio>
#include "queue.h"
using namespace std;

//Alternatif 1
Tabnine | Edit | Test | Explain | Document
void createQueue(Queue &Q) {
    Q.head = -1;
    Q.tail = -1;
}

Tabnine | Edit | Test | Explain | Document
bool isEmptyQueue(Queue Q) {
    return (Q.head == -1 && Q.tail == -1);
}

Tabnine | Edit | Test | Explain | Document
bool isFullQueue(Queue Q) {
    return (Q.tail == MAX_QUEUE - 1);
}

Tabnine | Edit | Test | Explain | Document
void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) {
        cout << "Queue penuh!" << endl;
        return;
    }

    if (isEmptyQueue(Q)) {
        Q.head = 0;
        Q.tail = 0;
```

```
29      } else {
30          Q.tail++;
31      }
32
33      Q.info[Q.tail] = x;
34  }
35
    Tabnine | Edit | Test | Explain | Document
36  infotype dequeue(Queue &Q) {
37      if (isEmptyQueue(Q)) {
38          cout << "Queue kosong!" << endl;
39          return -1;
40      }
41
42      infotype x = Q.info[Q.head];
43
44      if (Q.head == Q.tail) {
45          Q.head = Q.tail = -1;
46      } else {
47          for (int i = Q.head; i < Q.tail; i++) {
48              Q.info[i] = Q.info[i + 1];
49          }
50          Q.tail--;
51      }
52
53      return x;
54  }
55
```

```
56  //Alternatif 2
    Tabnine | Edit | Test | Explain | Document
57  void enqueue2(Queue &Q, infotype x) {
58      if (Q.tail == MAX_QUEUE - 1) {
59          cout << "Queue penuh (Alt2)!" << endl;
60          return;
61      }
62
63      if (isEmptyQueue(Q)) {
64          Q.head = 0;
65          Q.tail = 0;
66      } else {
67          Q.tail++;
68      }
69
70      Q.info[Q.tail] = x;
71  }
72
    Tabnine | Edit | Test | Explain | Document
73  infotype dequeue2(Queue &Q) {
74      if (isEmptyQueue(Q)) {
75          cout << "Queue kosong (Alt2)!" << endl;
76          return -1;
77      }
78
79      infotype x = Q.info[Q.head];
80
81      if (Q.head == Q.tail) {
82          Q.head = Q.tail = -1;
83      } else {
84          Q.head++;
85      }
```

```
86
87      return x;
88  }
89
90  //Alternatif 3
    Tabnine | Edit | Test | Explain | Document
91  void createQueue3(Queue &Q) {
92      Q.head = -1;
93      Q.tail = -1;
94  }
95
    Tabnine | Edit | Test | Explain | Document
96  bool isFullQueue3(Queue Q) {
97      return ((Q.tail + 1) % MAX_QUEUE) == Q.head;
98  }
99
    Tabnine | Edit | Test | Explain | Document
100 void enqueue3(Queue &Q, infotype x) {
101     if (isFullQueue3(Q)) {
102         cout << "Queue penuh (Alt3)!" << endl;
103         return;
104     }
105
106     if (isEmptyQueue(Q)) {
107         Q.head = 0;
108         Q.tail = 0;
109     } else {
110         Q.tail = (Q.tail + 1) % MAX_QUEUE;
111     }
112
```

```
113     Q.info[Q.tail] = x;
114 }
115
    Tabnine | Edit | Test | Explain | Document
116 infotype dequeue3(Queue &Q) {
117     if (isEmptyQueue(Q)) {
118         cout << "Queue kosong (Alt3)!" << endl;
119         return -1;
120     }
121
122     infotype x = Q.info[Q.head];
123
124     if (Q.head == Q.tail) {
125         Q.head = Q.tail = -1;
126     } else {
127         Q.head = (Q.head + 1) % MAX_QUEUE;
128     }
129
130     return x;
131 }
132
    Tabnine | Edit | Test | Explain | Document
133 void printInfo(Queue Q) {
134     printf("%-3d - %-3d | ", Q.head, Q.tail);
135
136     if (isEmptyQueue(Q)) {
137         cout << "empty queue";
138     } else {
139         for (int i = Q.head; i != Q.tail; i = (i + 1) % MAX_QUEUE) {
140             printf("%-3d", Q.info[i]);
141         }
142         printf("%-3d", Q.info[Q.tail]);
143     }
144
145     cout << endl;
146 }
147
```

(file main.cpp)

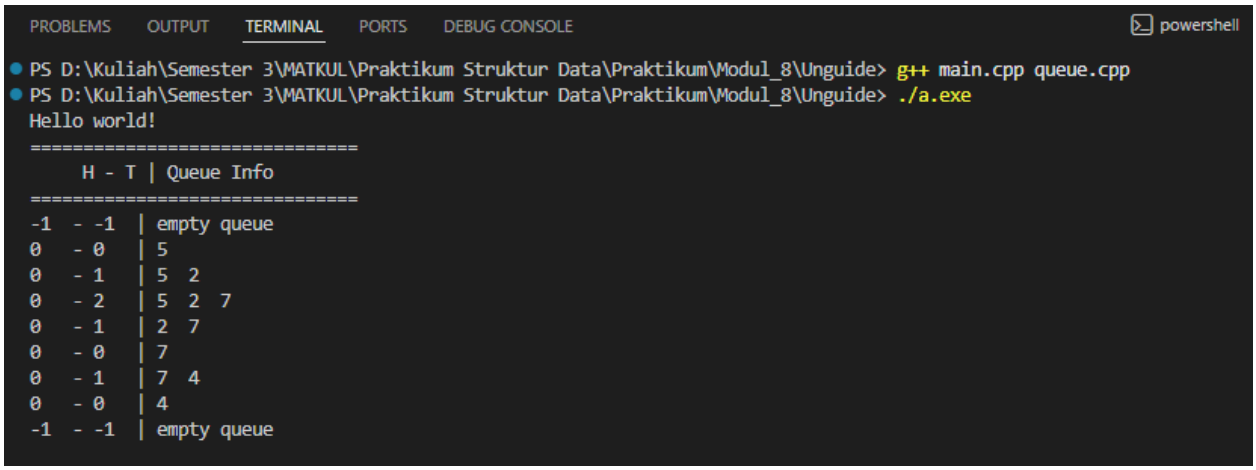
main.cpp > main()

```cpp
#include <iostream>
#include "queue.h"
using namespace std;

Tabnine | Edit | Test | Explain | Document
int main() {

    cout << "Hello world!" << endl;
    cout << "==============================" << endl;
    cout << "     H - T | Queue Info" << endl;
    cout << "==============================" << endl;

    Queue Q1;
    createQueue(Q1);

    printInfo(Q1);

    enqueue(Q1, 5); printInfo(Q1);
    enqueue(Q1, 2); printInfo(Q1);
    enqueue(Q1, 7); printInfo(Q1);
    dequeue(Q1);    printInfo(Q1);
    dequeue(Q1);    printInfo(Q1);
    enqueue(Q1, 4); printInfo(Q1);
    dequeue(Q1);    printInfo(Q1);
    dequeue(Q1);    printInfo(Q1);

    cout << endl << endl;


    cout << "========================================" << endl;
    cout << "  QUEUE ALTERNATIF 2 (HEAD BERGERAK)" << endl;
    cout << "========================================" << endl;
```

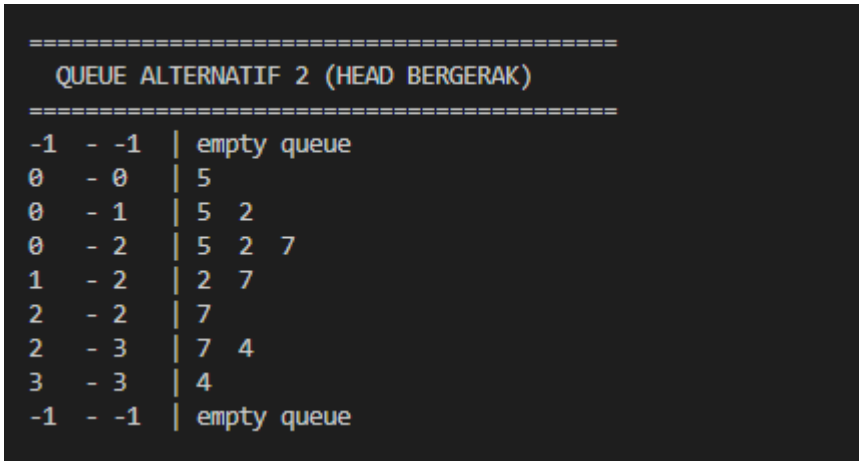
```
32
33      Queue Q2;
34      createQueue(Q2);
35
36      printInfo(Q2);
37      enqueue2(Q2, 5); printInfo(Q2);
38      enqueue2(Q2, 2); printInfo(Q2);
39      enqueue2(Q2, 7); printInfo(Q2);
40
41      dequeue2(Q2);    printInfo(Q2);
42      dequeue2(Q2);    printInfo(Q2);
43
44      enqueue2(Q2, 4); printInfo(Q2);
45      dequeue2(Q2);    printInfo(Q2);
46      dequeue2(Q2);    printInfo(Q2);
47
48      cout << endl << endl;
49
50
51      cout << "========================================" << endl;
52      cout << "    QUEUE ALTERNATIF 3 (CIRCULAR QUEUE)" << endl;
53      cout << "========================================" << endl;
54
55      Queue Q3;
56      createQueue3(Q3);
57
58      printInfo(Q3);
59      enqueue3(Q3, 5); printInfo(Q3);
60      enqueue3(Q3, 2); printInfo(Q3);
61      enqueue3(Q3, 7); printInfo(Q3);
62      enqueue3(Q3, 8); printInfo(Q3);
63      enqueue3(Q3, 9); printInfo(Q3);
64
65      dequeue3(Q3); printInfo(Q3);
66      enqueue3(Q3, 4); printInfo(Q3);   // tail berputar
67      dequeue3(Q3); printInfo(Q3);
68      dequeue3(Q3); printInfo(Q3);
69      dequeue3(Q3); printInfo(Q3);
70
71      return 0;
72  }
73
```

Screenshots Output

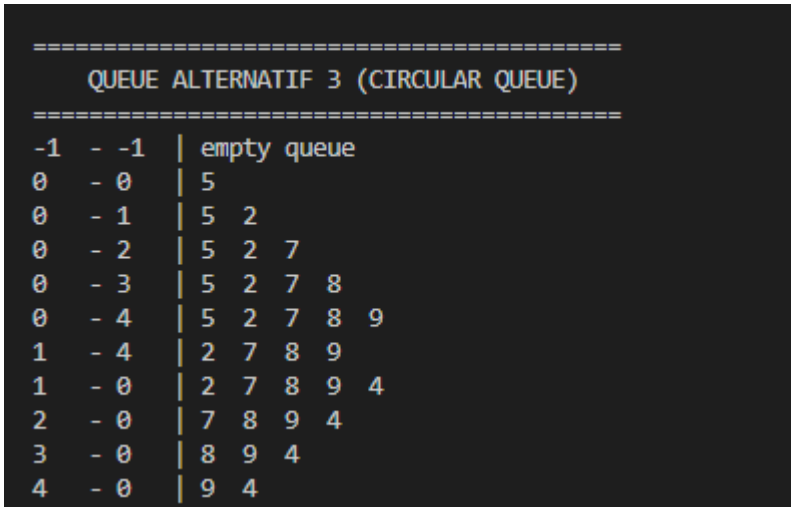
1) Output Queue (head diam, tail bergerak)



2) Output Queue (head bergerak, tail bergerak)



3) Output Queue (head dan tail berputar)



Deskripsi:

Program di atas merupakan implementasi struktur data queue (antrian) menggunakan bahasa C++ dengan tiga pendekatan berbeda, yaitu Alternatif 1 (head diam, tail bergerak), Alternatif 2 (head dan tail bergerak), dan Alternatif 3 (circular queue). Struktur Queue memiliki array info untuk menyimpan elemen serta dua indeks, yaitu head sebagai posisi elemen depan dan tail sebagai posisi elemen belakang. Pada Alternatif 1, operasi dequeue akan menggeser seluruh elemen ke depan setiap kali data diambil. Alternatif 2 memperbaiki kelemahan tersebut dengan menggeser pointer head tanpa memindahkan isi array. Alternatif 3 menggunakan konsep perputaran indeks dengan operasi modulo sehingga ruang array dapat digunakan sepenuhnya meskipun tail sudah mencapai akhir. Program menyediakan fungsi enqueue, dequeue, dan printInfo pada setiap alternatif untuk menampilkan perubahan queue. Pada fungsi main, program mendemonstrasikan setiap versi dengan memasukkan data 5, 2, 7, kemudian melakukan beberapa operasi dequeue dan enqueue. Pada Alternatif 3 misalnya, setelah queue berisi [5, 2, 7, 8, 9] dan dilakukan dequeue, kemudian ditambahkan elemen baru 4, posisi tail akan berputar kembali ke indeks awal, menunjukkan keunggulan circular queue dalam memanfaatkan ruang array secara efisien.

D. Kesimpulan

Berdasarkan hasil praktikum pada Modul VIII tentang Struktur Data Queue, dapat disimpulkan bahwa queue merupakan struktur data linier yang bekerja dengan prinsip FIFO (First In, First Out), yaitu elemen yang pertama kali masuk akan menjadi elemen pertama yang dikeluarkan. Melalui praktikum ini, mahasiswa memahami implementasi dasar queue menggunakan array dan linked list, serta variasi pendekatan seperti head diam–tail bergerak, head dan tail bergerak, hingga circular queue yang lebih efisien dalam pemanfaatan memori. Operasi penting seperti enqueue, dequeue, isEmpty, isFull, dan printInfo dipraktikkan untuk menunjukkan cara kerja antrian dalam pemrosesan data berurutan. Sebagai contoh, ketika pengguna memasukkan data 5, 2, dan 7, maka isi queue menjadi: [HEAD] 5 | 2 | 7 [TAIL]; setelah dilakukan dequeue, elemen yang keluar adalah 5, sesuai prinsip FIFO, dan ketika ditambahkan angka 4 pada circular queue, posisi tail berputar sehingga antrian dapat terus digunakan tanpa perlu menggeser elemen. Secara keseluruhan, praktikum ini memperkuat pemahaman mahasiswa mengenai konsep antrian, efisiensi struktur memori, serta penerapan queue pada berbagai aplikasi seperti penjadwalan proses, manajemen layanan, dan pemrosesan data berurutan.

E. Referensi

Dicoding. (2023). Struktur Data Queue: Pengertian, Fungsi, dan Jenisnya. Diakses dari https://dicoding.com/blog/struktur-data-queue

Sitanggang, B. F., Tinambunan, D., & Sinaga, J. H. (2021). Jurnal struktur data: Menyelesaikan permasalahan pengolahan data menggunakan metode queue (antrean) [Tugas Mata Kuliah, STIKOM Tunas Bangsa Pematangsiantar]. Scribd. https://www.scribd.com/document/511997415/Jurnal-Struktur-Data-MENYELESAIKAN-PERMASALAHAN-PENGOLAHAN-DATA-MENGGUNAKAN-METODE-QUEUE

Telkom University. (n.d.). Queue: Fungsi, Jenis, dan Implementasi dalam Pemrograman. Diakses dari https://share.google/UG9UTFTYqow0fZOHJ