

LAPORAN PRAKTIKUM

Modul VII

Queue



Disusun oleh:

Muhammad Irsyad : **2211102048**

Dosen

Wahyu Andi Saputra, S.Pd., M.Eng

**PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2023**

BAB I

Tujuan Pembelajaran

1. Mahasiswa mampu menjelaskan definisi dan konsep dari Double Queue
2. Mahasiswa mampu menerapkan operasi tamba, menghapus pada Queue
3. Mahasiswa mampu menerapkan operasi tampil data pada queue

BAB II

Dasar Teori

Pengertian

Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode FIFO (First-In First-Out). Data yang pertama dimasukkan ke dalam queue akan menjadi data yang pertama pula untuk dikeluarkan dari queue. Queue mirip dengan konsep antrian pada kehidupan sehari-hari.

Implementasi

Implementasi queue dapat dilakukan dengan menggunakan array atau linked list. Struktur data queue terdiri dari dua pointer yaitu front dan rear. front adalah pointer ke elemen pertama dalam queue dan rear adalah pointer ke elemen terakhir dalam queue.

Fungsi-fungsi pada Queue

Pada Queue, karena perubahan data selalu mengacu pada Head, maka hanya ada 1 jenis insert maupun delete. Prosedur ini sering disebut Enqueue dan Dequeue pada kasus Queue. Untuk Enqueue, cukup tambahkan elemen setelah elemen terakhir Queue, dan untuk Dequeue, cukup "geser"kan Head menjadi elemen selanjutnya.

Operasi pada Queue

`enqueue()`: menambahkan data ke dalam queue.

`dequeue()`: mengeluarkan data dari queue.

`peek()`: mengambil data dari queue tanpa menghapusnya.

`isEmpty()`: mengecek apakah queue kosong atau tidak.

`isFull()`: mengecek apakah queue penuh atau tidak.

`size()`: menghitung jumlah elemen dalam queue.

BAB III

LATIHAN KELAS – GUIDED

Guided 1

Source Code

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

// queue array
int maksimalQueue = 5; // maksimal antrian
int front = 0;         // penanda antrian
int back = 0;          // penanda
string queueTeller[5]; // fungsi pengecekan
bool isFull()
{ // pengecekan antrian penuh atau tidak
    if (back == maksimalQueue)
    {
        return true; //=1
    }
    else
    {
        return false;
    }
}
// fungsi pengecekan
bool isEmpty()
{ // antriannya kosong atau tidak
    if (back == 0)
    {
        return true;
    }
    else
    {
```

```

        return false;
    }
}
// fungsi menambahkan antrian
void enqueueAntrian(string data)
{
    if (isFull())
    {
        cout << "antrian penuh" << endl;
    }
    else
    {
        if (isEmpty()) // kondisi ketika queue kosong
        {
            queueTeller[0] = data;
            front++; // front = front+1;
            back++;
        }
        else
        {
            // antriannya ada isi
            queueTeller[back] = data; // queueTeller[1]=data
            back++; // back=back+1;2
        }
    }
}
// fungsi mengurangi antrian
void dequeueAntrian()
{
    if (isEmpty())
    {
        cout << "antrian kosong" << endl;
    }
    else
    {

```

```

        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = queueTeller[i + 1];
        }
        back--;
    }
}

// fungsi menghitung banyak antrian
int countQueue()
{
    return back;
}

// fungsi menghapus semua antrian
void clearQueue()
{
    if (isEmpty())
    {
        cout << "antrian kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = "";
        }
        back = 0;
        front = 0;
    }
}

// fungsi melihat antrian
void viewQueue()
{
    cout << "data antrian teller : " << endl;
    for (int i = 0; i < maksimalQueue; i++)

```

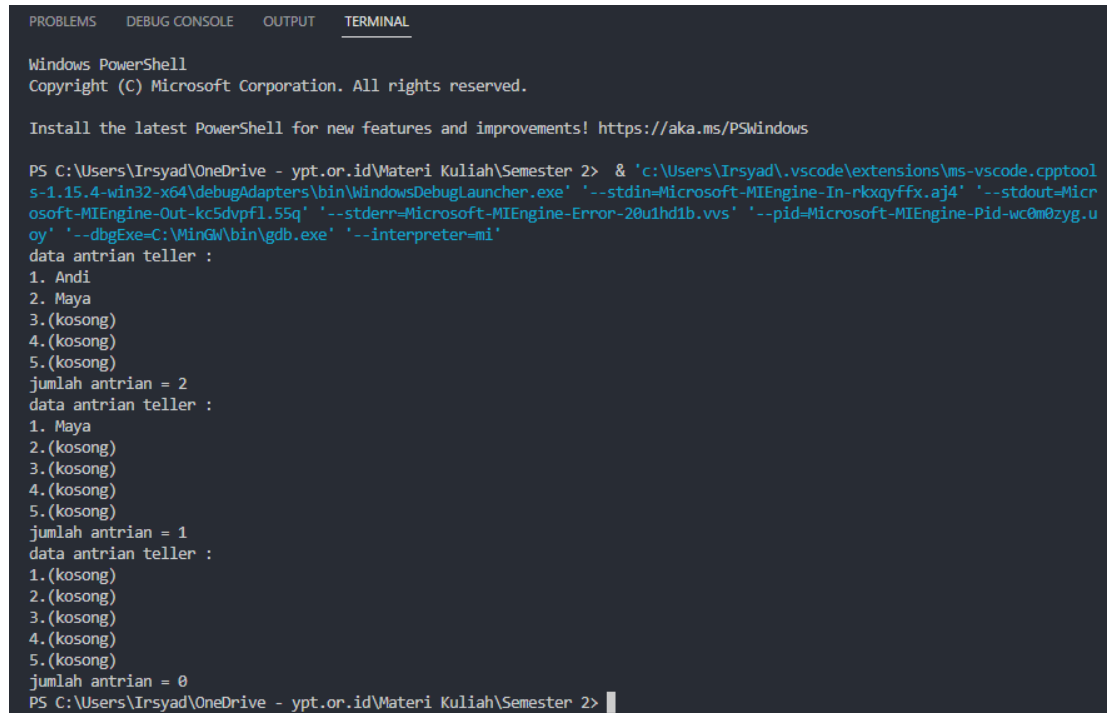
```

    {
        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;
}

```

Screenshot Program



```
PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Irsyad\OneDrive - ypt.or.id\Materi Kuliah\Semester 2> & 'c:\Users\Irsyad\.vscode\extensions\ms-vscode.cpptools-1.15.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-rkxqyffx.aj4' '--stdout=Microsoft-MIEngine-Out-kc5dvpfl.55q' '--stderr=Microsoft-MIEngine-Error-20u1hd1b.vvs' '--pid=Microsoft-MIEngine-Pid-wc0m0zyg.uoy' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'
data antrian teller :
1. Andi
2. Maya
3.(kosong)
4.(kosong)
5.(kosong)
jumlah antrian = 2
data antrian teller :
1. Maya
2.(kosong)
3.(kosong)
4.(kosong)
5.(kosong)
jumlah antrian = 1
data antrian teller :
1.(kosong)
2.(kosong)
3.(kosong)
4.(kosong)
5.(kosong)
jumlah antrian = 0
PS C:\Users\Irsyad\OneDrive - ypt.or.id\Materi Kuliah\Semester 2> |
```

Deskripsi Program

Kode program di atas adalah implementasi sederhana dari struktur data antrian (queue) menggunakan array dalam bahasa pemrograman C++.

1. Baris 1-3: Mendefinisikan header file yang digunakan dan menggunakan namespace `std`.
2. Baris 6-10: Mendefinisikan beberapa variabel dan array yang digunakan dalam implementasi antrian. `maksimalQueue` adalah ukuran maksimal antrian, `front` dan `back` adalah penanda posisi depan dan belakang antrian, dan `queueTeller` adalah array yang digunakan untuk menyimpan data antrian.
3. Baris 12-17: Definisi fungsi `isFull()`. Fungsi ini digunakan untuk memeriksa apakah antrian penuh atau tidak. Jika `back` sama dengan `maksimalQueue`, berarti antrian penuh dan fungsi mengembalikan nilai `true`, jika tidak, fungsi mengembalikan nilai `false`.

4. Baris 19-25: Definisi fungsi ``isEmpty()``. Fungsi ini digunakan untuk memeriksa apakah antrian kosong atau tidak. Jika ``back`` sama dengan 0, berarti antrian kosong dan fungsi mengembalikan nilai ``true``, jika tidak, fungsi mengembalikan nilai ``false``.
5. Baris 27-40: Definisi fungsi ``enqueueAntrian(string data)``. Fungsi ini digunakan untuk menambahkan data ke antrian. Jika antrian penuh, akan ditampilkan pesan "antrian penuh". Jika antrian kosong, data ditambahkan pada posisi pertama dan ``front`` dan ``back`` diinkremenkan. Jika antrian tidak kosong, data ditambahkan pada posisi ``back`` dan ``back`` diinkremenkan.
6. Baris 42-52: Definisi fungsi ``dequeueAntrian()``. Fungsi ini digunakan untuk menghapus data dari antrian. Jika antrian kosong, akan ditampilkan pesan "antrian kosong". Jika antrian tidak kosong, data pada posisi pertama (indeks 0) dihapus dengan menggeser seluruh elemen antrian ke posisi sebelumnya dan ``back`` dikurangi satu.
7. Baris 54-57: Definisi fungsi ``countQueue()``. Fungsi ini digunakan untuk menghitung jumlah elemen dalam antrian. Fungsi ini mengembalikan nilai ``back``, yaitu jumlah elemen yang ada dalam antrian.
8. Baris 59-69: Definisi fungsi ``clearQueue()``. Fungsi ini digunakan untuk menghapus semua elemen dalam antrian. Jika antrian kosong, akan ditampilkan pesan "antrian kosong". Jika antrian tidak kosong, semua elemen dalam array ``queueTeller`` dikosongkan, ``back`` dan ``front`` diset ke 0.
9. Baris 71-84: Definisi fungsi ``viewQueue()``. Fungsi ini digunakan untuk menampilkan data dalam antrian. Fungsi ini melakukan iterasi melalui array ``queueTeller`` dan menampilkan nomor antrian dan data pada setiap posisi. Jika posisi tersebut kosong, maka akan ditampilkan "(kosong)".
10. Baris 86-101: Fungsi ``main()``. Fungsi utama program ini. Pada bagian ini, dilakukan pemanggilan beberapa fungsi antrian untuk menambah, menghapus, dan melihat data dalam antrian. Pertama, dilakukan penambahan dua data menggunakan fungsi ``enqueueAntrian()``. Kemudian, dilakukan pemanggilan

fungsi `viewQueue()` untuk menampilkan data dalam antrian. Setelah itu, dilakukan pemanggilan fungsi `countQueue()` untuk menghitung jumlah elemen dalam antrian. Selanjutnya, dilakukan pemanggilan fungsi `dequeueAntrian()` untuk menghapus data dari antrian dan kembali menampilkan data dalam antrian. Kemudian, dilakukan pemanggilan fungsi `countQueue()` lagi untuk menghitung jumlah elemen dalam antrian setelah penghapusan. Terakhir, dilakukan pemanggilan fungsi `clearQueue()` untuk menghapus semua data dalam antrian dan kembali menampilkan data dalam antrian. Akhirnya, dilakukan pemanggilan fungsi `countQueue()` lagi untuk menghitung jumlah elemen dalam antrian setelah penghapusan semua data.

TUGAS – UNGUIDED

Unguided 1

Source Code

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

struct Node
{
    string nama;
    string nim;
    Node *next;
};

class AntrianMahasiswa
{
private:
    Node *front;
    Node *rear;

public:
    AntrianMahasiswa()
```

```

{
    front = nullptr;
    rear = nullptr;
}

bool isEmpty()
{
    return (front == nullptr);
}

void enqueueData(string nama, string nim)
{
    Node *newNode = new Node;
    newNode->nama = nama;
    newNode->nim = nim;
    newNode->next = nullptr;

    if (isEmpty())
    {
        front = newNode;
        rear = newNode;
    }
    else
    {
        rear->next = newNode;
        rear = newNode;
    }

    cout << "Data " << nama << " (" << nim << ") ditambahkan" <<
endl;
}

void dequeueData()
{

```

```

        if (isEmpty())
        {
            cout << "Antrian kosong" << endl;
        }
        else
        {
            Node *temp = front;
            front = front->next;
            cout << "Data " << temp->nama << " (" << temp->nim << ")
dihapus" << endl;
            delete temp;

            if (front == nullptr)
            {
                rear = nullptr;
            }
        }
    }

void viewData()
{
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        cout << "Data mahasiswa dalam antrian: " << endl;
        Node *current = front;
        int i = 1;

        while (current != nullptr)
        {

```

```

        cout << i << ". Nama: " << current->nama << ", NIM:
" << current->nim << endl;
        current = current->next;
        i++;
    }
}

void resetData()
{
    while (!isEmpty())
    {
        dequeueData();
    }
}

};

int main()
{
    AntrianMahasiswa antrian;

    int pilihan;
    string nama, nim;

    do
    {
        cout << "Menu: " << endl;
        cout << "1. Masukkan Data" << endl;
        cout << "2. Hapus Satu Data" << endl;
        cout << "3. Reset Data" << endl;
        cout << "4. Tampil Data" << endl;
        cout << "5. Keluar" << endl;
        cout << "Pilihan Anda: ";
        cin >> pilihan;
    }
}

```

```
switch (pilihan)
{
case 1:
    cout << "Masukkan Nama Mahasiswa: ";
    cin.ignore();
    getline(cin, nama);
    cout << "Masukkan NIM Mahasiswa: ";
    cin >> nim;
    antrian.enqueueData(nama, nim);
    break;

case 2:
    antrian.dequeueData();
    break;

case 3:
    antrian.resetData();
    break;

case 4:
    antrian.viewData();
    break;

case 5:
    cout << "Terima kasih!" << endl;
    break;

default:
    cout << "Pilihan tidak valid!" << endl;
    break;
}

cout << endl;
```

```

    } while (pilihan != 5);

    return 0;
}

```

Screenshot Program

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Irsyad\OneDrive - ypt.or.id\Materi Kuliah\Semester 2> & 'c:\Users\Irsyad\.vscode\extensions\ms-vscode.cpptools-1.15.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-q4pbu2aq.ek3' '--stdout=Microsoft-MIEngine-Out-sjnd125q.3pt' '--stderr=Microsoft-MIEngine-Error-mmbwboik.hke' '--pid=Microsoft-MIEngine-Pid-2v5wa1ml.vq4' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'
Menu:
1. Masukkan Data
2. Hapus Satu Data
3. Reset Data
4. Tampil Data
5. Keluar
Pilihan Anda: 1
Masukkan Nama Mahasiswa: Muhammad Irsyad
Masukkan NIM Mahasiswa: 2211102048
Data Muhammad Irsyad (2211102048) ditambahkan

Menu:
1. Masukkan Data
2. Hapus Satu Data
3. Reset Data
4. Tampil Data
5. Keluar
Pilihan Anda: 4
Data mahasiswa dalam antrian:
1. Nama: Muhammad Irsyad, NIM: 2211102048

Menu:
1. Masukkan Data
2. Hapus Satu Data
3. Reset Data
4. Tampil Data
5. Keluar
Pilihan Anda: █

```

Deskripsi Program

1. Deklarasi struktur `Node` yang memiliki dua anggota data bertipe string (`nama` dan `nim`) serta pointer `next` yang menunjukkan ke node berikutnya.
2. Deklarasi kelas `AntrianMahasiswa` dengan anggota data `front` dan `rear` bertipe pointer ke `Node`, yang menunjukkan node terdepan dan terbelakang dalam antrian.
3. Pada konstruktor kelas `AntrianMahasiswa`, menginisialisasi `front` dan `rear` dengan `nullptr` untuk menandakan antrian kosong.

4. Fungsi ``isEmpty()`` digunakan untuk memeriksa apakah antrian kosong atau tidak.
5. Fungsi ``enqueueData(string nama, string nim)`` digunakan untuk menambahkan data mahasiswa baru ke dalam antrian. Fungsi ini membuat node baru, mengisi data dengan ``nama`` dan ``nim`` yang diberikan, dan menambahkannya ke akhir antrian. Jika antrian kosong, maka ``front`` dan ``rear`` menunjuk ke node baru tersebut.
6. Fungsi ``dequeueData()`` digunakan untuk menghapus data mahasiswa dari depan antrian. Fungsi ini menghapus node terdepan (`front`) dari antrian, memperbarui ``front`` ke node berikutnya, dan menghapus node yang dihapus dari memori.
7. Fungsi ``viewData()`` digunakan untuk menampilkan data mahasiswa yang ada dalam antrian. Fungsi ini melakukan iterasi melalui setiap node, mulai dari ``front`` hingga ``rear``, dan menampilkan nama dan nim setiap mahasiswa.
8. Fungsi ``resetData()`` digunakan untuk menghapus semua data dalam antrian dengan memanggil fungsi ``dequeueData()`` secara berulang hingga antrian kosong.
9. Fungsi ``main()`` adalah fungsi utama program. Pada fungsi ini, terdapat loop `do-while` yang berjalan hingga pilihan pengguna adalah 5 (keluar).
10. Pada setiap iterasi loop, program menampilkan menu dan meminta pengguna untuk memasukkan pilihan.
11. Berdasarkan pilihan yang dimasukkan pengguna, program memanggil fungsi-fungsi yang sesuai dari objek ``antrian``.
12. Pilihan 1 meminta pengguna memasukkan nama dan nim mahasiswa baru, kemudian memanggil fungsi ``enqueueData()`` untuk menambahkan data tersebut ke antrian.
13. Pilihan 2 memanggil fungsi ``dequeueData()`` untuk menghapus data mahasiswa dari depan antrian.

14. Pilihan 3 memanggil fungsi ``resetData()`` untuk menghapus semua data dalam antrian.
15. Pilihan 4 memanggil fungsi ``viewData()`` untuk menampilkan data mahasiswa dalam antrian.
16. Pilihan 5 keluar dari loop dan program berakhir.

BAB IV

KESIMPULAN

1. Queue adalah struktur data linier yang mengikuti prinsip FIFO (First-In-First-Out), di mana elemen yang pertama dimasukkan ke dalam antrian adalah yang pertama kali dihapus.
2. Implementasi dasar queue dapat dilakukan menggunakan array atau linked list. Dalam kode yang telah diberikan, kita dapat melihat contoh implementasi queue menggunakan array dan kemudian diubah menjadi menggunakan linked list.
3. Operasi utama dalam queue adalah enqueue (memasukkan elemen ke dalam antrian) dan dequeue (menghapus elemen dari antrian). Operasi enqueue dilakukan pada akhir antrian, sementara operasi dequeue dilakukan pada elemen pertama antrian.
4. Pengecekan apakah antrian penuh atau kosong sangat penting dalam implementasi queue. Dalam kode yang telah diberikan, terdapat fungsi ``isFull()`` untuk memeriksa apakah antrian penuh dan ``isEmpty()`` untuk memeriksa apakah antrian kosong.
5. Selain itu, terdapat pula operasi lain seperti menghitung jumlah elemen dalam antrian (``countQueue()``) dan menghapus semua elemen dalam antrian (``clearQueue()``).
6. Queue sering digunakan dalam situasi di mana elemen harus diakses dalam urutan yang spesifik, seperti pengolahan antrian pelanggan, penjadwalan tugas, atau simulasi proses.

7. Dalam kode yang telah diberikan, konsep antrian diterapkan dengan atribut "Nama Mahasiswa" dan "NIM Mahasiswa" menggunakan linked list. Pengguna dapat memasukkan data, menghapus satu data, mengatur ulang data, dan menampilkan data melalui menu yang disediakan.
8. Memahami konsep dan implementasi queue penting dalam pemrograman karena memungkinkan pengolahan data dengan mempertahankan urutan yang sesuai dan mengoptimalkan efisiensi proses.

Dengan memahami konsep dan operasi dalam queue, kita dapat mengaplikasikannya dalam berbagai situasi di mana diperlukan pengelolaan data berdasarkan prinsip FIFO.