

Tugas Pendahuluan Modul 8

STRUKTUR DATA - Ganjil 2024/2025

"08_Queue"

Ketentuan Tugas Pendahuluan

1. Tugas Pendahuluan dikerjakan secara **Individu**.
2. TP ini bersifat **WAJIB**, tidak mengerjakan = **PENGURANGAN POIN JURNAL / TES ASESMEN**.
3. Hanya **MENGUMPULKAN** tetapi **TIDAK MENGERJAKAN** = **PENGURANGAN POIN JURNAL / TES ASESMEN**.
4. Deadline pengumpulan TP Modul 2 adalah Senin, 30 September 2024 pukul 07.30 WIB.
5. **TIDAK ADA TOLERANSI KETERLAMBATAN, TERLAMBAT ATAU TIDAK MENGUMPULKAN TP MAKA DIANGGAP TIDAK MENGERJAKAN**.
6. **DILARANG PLAGIAT (PLAGIAT = E)**.
7. Kerjakan TP dengan jelas agar dapat dimengerti.
8. Codingan diupload di Github dan upload Laporan di Lab menggunakan format **PDF** dengan ketentuan:
TP_MOD_[XX]_NIM_NAMA.pdf

CP (WA):

- Andini (082243700965)
- Imelda (082135374187)

SELAMAT MENGERJAKAN^^

LAPORAN PRAKTIKUM
PERTEMUAN 8
STRUKTUR DATA



Nama :

Zulfa Mustafa Akhyar Iswahyudi (2311104010)

Dosen :

Yudha Islami Sulistya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

A. Tujuan

Untuk melatih kompetensi Mahasiswa untuk memperdalam skill pemrograman C++

B. Tools

Codeblocks, VSCode, Github

LATIHAN – UNGUIDED

Dari Kodingan yang dibuat hari senin minggu ke-8 bersama Asprak, didapatkan hasil kodingan berikut :

guided.cpp

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

const int maksimalQueue = 5;
int front = 0;
int back = 0;
string queueTeller[5];

bool isFull()
{
    if (back == maksimalQueue)
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool isEmpty()
{
    if (back == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}

void enqueueAntrian(string data)
{
    if (isFull())
    {
        cout << "Antrian sudah penuh" << endl;
    }
    else
    {
        if (isEmpty())
        {
            queueTeller[0] = data;
            back++;
            front++;
        }
        else
        {
            queueTeller[back] = data;
            back++;
        }
    }
}

void dequeueAntrian() // kurangi antrian data
{
    if (isEmpty())
    {
        cout << "Antrian sudah kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back - 1; i++) // pindahkan data kedepan
        {
            queueTeller[i] = queueTeller[i + 1];
        }
        queueTeller[back - 1] = ""; // kosongkan posisi terakhir
        back--;
    }
}

int countQueue()
{
    // Hitung total antrian data
    return back;
}

void clearQueue()
{
    // Menghapus Antrian
    if (isEmpty())
    {
        cout << "Antrian sudah kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = "";
        }
        back = 0;
        front = 0;
    }
}

void viewQueue()
{
    // Lihat 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;
}
```



2311104010_ZULFA MUSTAFA AKHYAR ISWAHYUDI
Han Goo-Rang

Sedangkan untuk pengayaan Unguided, diberikan tiga perintah berikut :

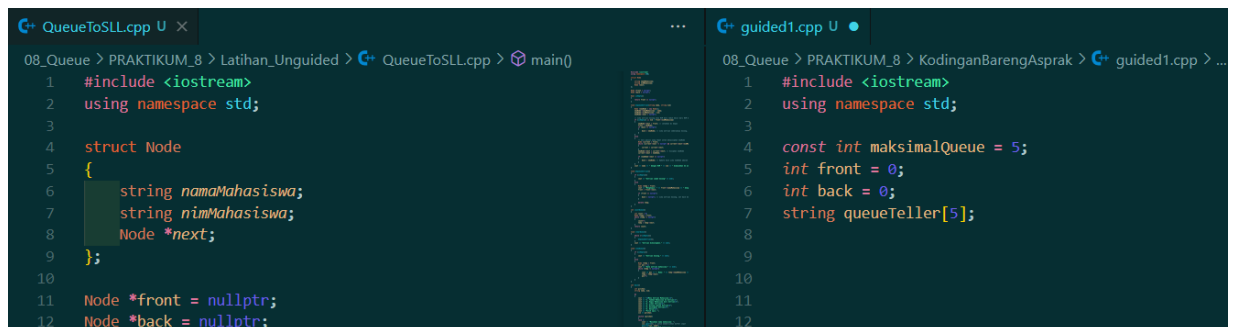
1. Ubahlah penerapan konsep queue pada bagian guided dari array menjadilinked list
2. Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIMMahasiswa
3. Modifikasi program pada soal 1 sehingga mahasiswa dapat diprioritaskan berdasarkan NIM (NIM yang lebih kecil didahulukan pada saat output).

Noted : Untuk data mahasiswa dan nim dimasukan oleh user

Dari ketiga soal, saya sudah ubah seluruh struktur kode program Queue menjadi basis SLL dan **mengubah variabel Queue menjadi dua variabel SLL yaitu 'Nama' dan 'NIM', serta mengaturnya agar program mengurutkan Nama/NIM Mahasiswa secara *ascending* (urut terkecil -> teratas)**

Berikut penjelasan perbandingan antar struktural program :

1.) Inisialisasi Awal



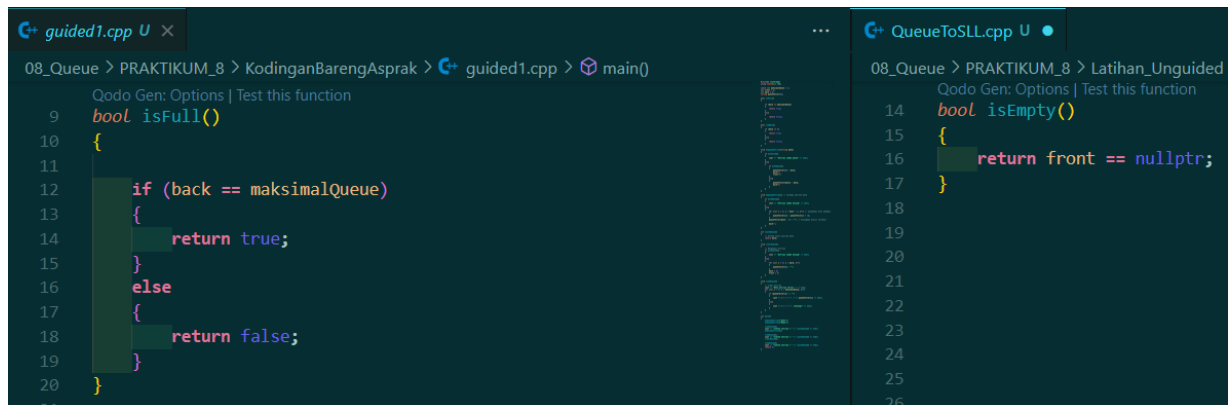
```
08_Queue > PRAKTIKUM_8 > Latihan_Unguided > QueueToSLL.cpp > main()
1  #include <iostream>
2  using namespace std;
3
4  struct Node
5  {
6      string namaMahasiswa;
7      string nimMahasiswa;
8      Node *next;
9  };
10
11 Node *front = nullptr;
12 Node *back = nullptr;
```

```
08_Queue > PRAKTIKUM_8 > KodanganBarengAsprak > guided1.cpp > ...
1  #include <iostream>
2  using namespace std;
3
4  const int maksimalQueue = 5;
5  int front = 0;
6  int back = 0;
7  string queueTeller[5];
8
9
10
11
12
```

Bisa kita lihat perbedaannya cukup banyak antara basis Queue dan SLL. Pada SLL, kita membuat struktur dari linkedlist yang berisikan tipe data dan inisialisasi Node *next agar data pada linkedlist dapat terus bertambah.

Sedangkan pada Queue, kita menginisialisasikan agar antrian berjumlah 5 elemen data dengan set default data 'front' dan data 'back' adalah nol. Buat deklarasi parameter 'queueTeller' dengan tipe data string nantinya.

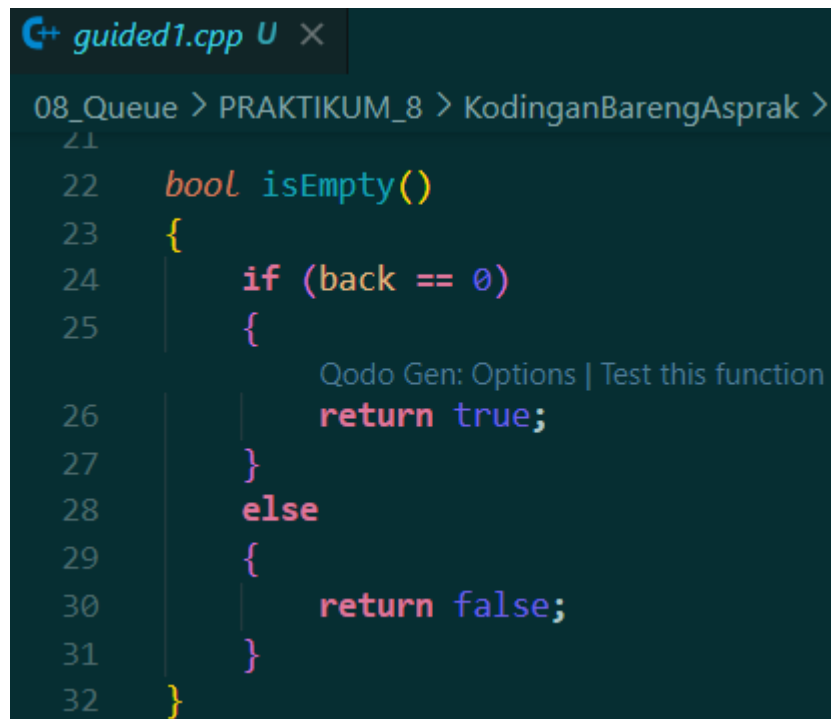
2.) Method True/False



```
08_Queue > PRAKTIKUM_8 > KodinganBarengAsprak > guided1.cpp > main()
Qodo Gen: Options | Test this function
9  bool isFull()
10 {
11
12     if (back == maksimalQueue)
13     {
14         return true;
15     }
16     else
17     {
18         return false;
19     }
20 }

08_Queue > PRAKTIKUM_8 > Latihan_Unguided
Qodo Gen: Options | Test this function
14 bool isEmpty()
15 {
16     return front == nullptr;
17 }
18
19
20
21
22
23
24
25
26
```

Seperti pada gambar, pada 'Queue' kita prioritaskan method boolean untuk mengecek kapasitas pada struktur Queue dengan pengkondisian nilai terakhir dengan konstanta. Sedangkan pada File Unguided 'QueueToSLL', kita cukup melakukan method True/False untuk mengecek apakah linkedlist kosong atau tidak dengan patokan data depan sebagai default-nya.



```
08_Queue > PRAKTIKUM_8 > KodinganBarengAsprak >
21
22 bool isEmpty()
23 {
24     if (back == 0)
25     {
26         Qodo Gen: Options | Test this function
27         return true;
28     }
29     else
30     {
31         return false;
32     }
33 }
```

Namun pada 'Queue' juga ada Method boolean untuk mengecek apakah struktur antrian kosong atau tidak. Jika data paling belakang dalam antrian adalah nol, maka akan bernilai true.

3.) Method Enqueue

The image shows two side-by-side code editors. The left editor, titled 'guided1.cpp', contains the implementation of the `enqueueAntrian` function for a standard queue. It uses an array `queueTeller` and checks `isFull()` and `isEmpty()` conditions. The right editor, titled 'QueueToSLL.cpp', contains the implementation of the `enqueueAntrian` function for a Singly Linked List (SLL). It uses a `Node` structure and checks `isEmpty()` and `nim < front->nimMahasiswa` conditions to insert a new node at the front or in the middle of the list.

```

34 void enqueueAntrian(string data)
35 {
36     if (isFull())
37     {
38         cout << "Antrian sudah penuh" << endl;
39     }
40     else
41     {
42         if (isEmpty())
43         {
44             queueTeller[0] = data;
45             back++;
46             front++;
47         }
48         else
49         {
50             queueTeller[back] = data;
51             back++;
52         }
53     }
54 }

19
20 void enqueueAntrian(string nama, string nim)
21 {
22     Node *newNode = new Node();
23     newNode->namaMahasiswa = nama;
24     newNode->nimMahasiswa = nim;
25     newNode->next = nullptr;
26
27     // Jika antrian kosong atau NIM baru Lebih kecil da
28     if (isEmpty() || nim < front->nimMahasiswa)
29     {
30         newNode->next = front; // Letakkan di depan
31         front = newNode;
32         if (back == nullptr)
33         {
34             back = newNode; // Jika antrian sebelumnya
35         }
36     }
37     else
38     {
39         // Cari posisi yang tepat untuk menyisipkan new
40         Node *current = front;
41         while (current->next != nullptr && current->nex
42         {
43             current = current->next;
44         }
45         newNode->next = current->next; // Sisipkan new
46         current->next = newNode;
47
48         if (newNode->next == nullptr)
49         {
50             back = newNode; // Update back jika new
51         }
52     }
53     cout << nama << " dengan NIM " << nim << " dimasukk
54 }

```

Pada Queue biasa method enqueue data diparameterkan dengan String data lalu lanjut ke pengkondisian fungsi `isFull` dan `isEmpty` yang bertujuan untuk pengecekan menyeluruh pada antrian yang akan diisikan data inputan.

Sementara pada SLL, method didefinisikan dengan `Node` yang membawa `newNode` untuk menginisialisasi variabel nama dan nim mahasiswa. Kemudian berlanjut pada pengecekan linkedlist beserta settingan `Node` pada pengkondisian 'while true' untuk memastikan bahwa data bisa terus bertambah selama pengkondisian bernilai benar.

4.) Method Dequeue

The image shows two side-by-side code editors. The left editor, titled 'guided1.cpp', contains the implementation of the `dequeueAntrian` function for a standard queue. It checks `isEmpty()` and shifts elements in the `queueTeller` array one position to the left. The right editor, titled 'QueueToSLL.cpp', contains the implementation of the `dequeueAntrian` function for a Singly Linked List (SLL). It checks `isEmpty()` and updates the `front` pointer to `front->next`, then deletes the `front` node.

```

56 void dequeueAntrian() // kurangi antrian data
57 {
58     if (isEmpty())
59     {
60         cout << "Antrian sudah kosong" << endl;
61     }
62     else
63     {
64         for (int i = 0; i < back - 1; i++) // pindahkan
65         {
66             queueTeller[i] = queueTeller[i + 1];
67         }
68         queueTeller[back - 1] = ""; // kosongkan posisi terakl
69
70         back--;
71     }
72 }

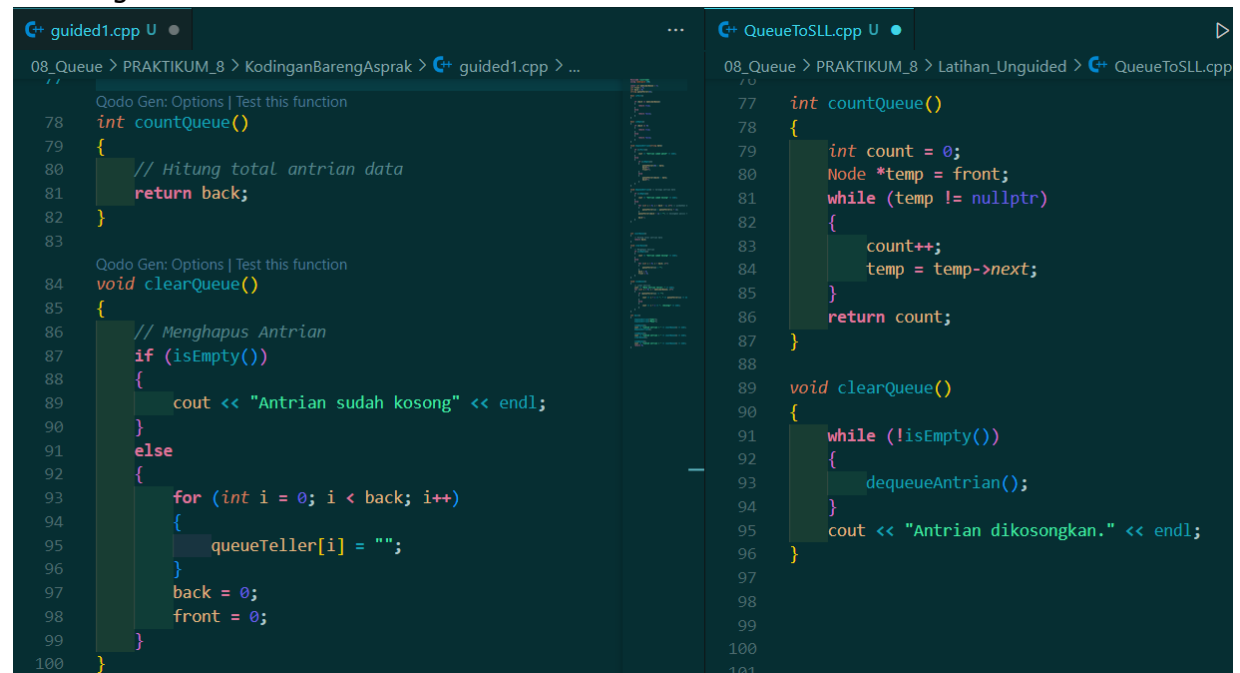
56 void dequeueAntrian()
57 {
58     if (isEmpty())
59     {
60         cout << "Antrian sudah kosong" << endl;
61     }
62     else
63     {
64         Node *temp = front;
65         cout << "Menghapus: " << front->namaMahasiswa
66         front = front->next;
67
68         if (front == nullptr)
69         {
70             back = nullptr; // Jika antrian kosong, se
71         }
72
73         delete temp;
74     }
75 }

```

Untuk Queue biasa method ini memberikan output bahwa antrian kosong pada function isEmpty. Untuk penghapusan data pada antrian dilakukan perulangan yang berjalan melalui data paling belakang dulu yang akan dihapus atau data terakhir yang masuk.

Sedangkan untuk linkedlist, untuk menghapus data dilakukan peng-iterasian yang akan mendeteksi data depan dan seterusnya sampai data paling belakang yang akan dideteksi terakhir.

5.) Counting



The image shows two side-by-side code editors. The left editor, titled 'guided1.cpp', contains the following code:

```
//  
Qodo Gen: Options | Test this function  
78 int countQueue()  
79 {  
80     // Hitung total antrian data  
81     return back;  
82 }  
83  
Qodo Gen: Options | Test this function  
84 void clearQueue()  
85 {  
86     // Menghapus Antrian  
87     if (isEmpty())  
88     {  
89         cout << "Antrian sudah kosong" << endl;  
90     }  
91     else  
92     {  
93         for (int i = 0; i < back; i++)  
94         {  
95             queueTeller[i] = "";  
96         }  
97         back = 0;  
98         front = 0;  
99     }  
100 }
```

The right editor, titled 'QueueToSLL.cpp', contains the following code:

```
//  
77 int countQueue()  
78 {  
79     int count = 0;  
80     Node *temp = front;  
81     while (temp != nullptr)  
82     {  
83         count++;  
84         temp = temp->next;  
85     }  
86     return count;  
87 }  
88  
89 void clearQueue()  
90 {  
91     while (!isEmpty())  
92     {  
93         dequeueAntrian();  
94     }  
95     cout << "Antrian dikosongkan." << endl;  
96 }  
97  
98  
99  
100  
101
```

Pada method khusus ini kita menghitung keseluruhan data antrian yang masuk. Pada Queue biasa kita hanya melakukan return sederhana. Sedangkan pada linkedlist, kita mendeklarasikan Node sebagai data sementara yang berada di posisi depan, kemudian dibuat looping 'while' selama linkedlist tidak kosong, data akan dikalkulasikan dari data paling depan hingga seterusnya.

Method Clear berguna untuk menghapus seluruh data baik pada Queue maupun linkedlist. Keduanya saling menggunakan konstanta yang sesuai function untuk melakukan pembersihan data.

6.) View


```

// guided1.cpp
102 void viewQueue()
103 {
104     // Lihat Antrian
105     cout << "Data antrian teller: " << endl;
106     for (int i = 0; i < maksimalQueue; i++)
107     {
108         if (queueTeller[i] != "")
109         {
110             cout << i + 1 << ". " << queueTeller[i] << endl;
111         }
112         else
113         {
114             cout << i + 1 << ". (kosong)" << endl;
115         }
116     }
117 }

// QueueToSLL.cpp
98 void viewQueue()
99 {
100     if (isEmpty())
101     {
102         cout << "Antrian kosong." << endl;
103     }
104     else
105     {
106         Node *temp = front;
107         int pos = 1;
108         cout << "Data antrian mahasiswa:" << endl;
109         while (temp != nullptr)
110         {
111             cout << pos << ". Nama: " << temp->namaMaha
112             temp = temp->next;
113             pos++;
114         }
115     }
116 }

```

Kedua struktur baik itu Queue dan linkedlist sama-sama melakukan pengkondisian untuk mengecek keseluruhan data yang diiterasikan secara runtut sesuai struktur masing-masing.

7.) Main Program

```

// guided1.cpp
122 int main()
123 {
124     enqueueAntrian("Andi");
125     enqueueAntrian("Maya");
126
127     viewQueue();
128     cout << "Jumlah antrian = " << countQueue() << endl;
129     dequeueAntrian();
130
131     viewQueue();
132     cout << "Jumlah antrian = " << countQueue() << endl;
133     clearQueue();
134
135     viewQueue();
136     cout << "Jumlah antrian = " << countQueue() << endl;
137     return 0;
138 }

// QueueToSLL.cpp
118 int main()
119 {
120     int pilihan;
121     string nama, nim;
122
123     do
124     {
125         cout << "\nMenu Antrian Mahasiswa:\n";
126         cout << "1. Tambah Mahasiswa ke Antrian\n";
127         cout << "2. Hapus Mahasiswa dari Antrian\n";
128         cout << "3. Lihat Antrian\n";
129         cout << "4. Hitung Jumlah Antrian\n";
130         cout << "5. Kosongkan Antrian\n";
131         cout << "6. Keluar\n";
132         cout << "Pilih opsi: ";
133         cin >> pilihan;
134
135         switch (pilihan)
136         {
137             case 1:
138                 cout << "Masukkan nama mahasiswa: ";
139                 cin.ignore(); // untuk membersihkan buffer
140                 getline(cin, nama);
141                 cout << "Masukkan NIM mahasiswa: ";
142                 cin >> nim;
143                 enqueueAntrian(nama, nim);
144                 break;
145
146             case 2:
147                 dequeueAntrian();
148                 break;
149
150             case 3:
151                 viewQueue();
152                 break;

```

Masing-masing Queue dan linkedlist berbeda jauh dalam pengoperasian programnya. Pada Queue kita melakukan inout data secara otomatis dengan method enqueue dan untuk menampilkan keseluruhan datanya menggunakan method viewQueue.

Sedangkan linkedlist, kita memasukkan inputan data secara manual yang dimana kita bisa memilih untuk Tambah, Hapus, Lihat, Hitung Total, Mengosongkan SLL, dan keluar. Disini kita buat percabangan masing-masing menu sesuai dengan function masing-masing method dengan 'switch case'.

Semoga Selalu diberi kemudahan^^