LAPORAN PRAKTIKUM

MODUL 8

QUEUE

Nama :

Fajar Budiawan (2311104039)

Dosen :

Yudha Islami Sulistya S.Kom, M.Cs

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

Unguided

1. Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list

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

// Definisikan struktur Node untuk linked list
struct Node {
    string data;    // Data untuk disimpan (bisa tipe lain, di sini string)
    Node* next;     // Pointer ke node berikutnya dalam list
};

// Definisikan kelas Queue untuk mengelola antrian berbasis linked list
class Queue {
private:
    Node* front;   // Pointer ke depan antrian
    Node* back;    // Pointer ke belakang antrian

public:
    // Konstruktor, inisialisasi queue dengan front dan back null
    Queue() : front(nullptr), back(nullptr) {}

    // Cek apakah antrian kosong
    bool isEmpty() {
        return front == nullptr;
    }

    // Menambahkan elemen ke dalam antrian (enqueue)
    void enqueue(const string& data) {
        Node* newNode = new Node{data, nullptr};  // Buat node baru

        if (isEmpty()) {
            front = back = newNode;  // Jika kosong, front dan back menunjuk ke node baru
        } else {
            back->next = newNode;  // Sambungkan node baru di belakang antrian
            back = newNode;        // Update pointer back ke node baru
        }
    }

    // Menghapus elemen dari antrian (dequeue)
    void dequeue() {
        if (isEmpty()) {
            cout << "Antrian Kosong" << endl;
            return;
        }

        Node* temp = front;  // Simpan node depan sementara
        front = front->next; // Pindahkan front ke node berikutnya
        if (front == nullptr) {
            back = nullptr;  // Jika antrian kosong, back juga null
        }
        delete temp;  // Hapus node lama (node yang terdepan)
    }

    // Menghitung jumlah elemen dalam antrian
    int count() {
        int count = 0;
        Node* current = front;
        while (current != nullptr) {
            count++;
            current = current->next;
        }
        return count;
    }

    // Menghapus semua elemen dalam antrian
    void clear() {
        while (!isEmpty()) {
            dequeue();  // Dequeue hingga antrian kosong
        }
    }

    // Menampilkan elemen-elemen dalam antrian
    void view() {
        if (isEmpty()) {
            cout << "Antrian Kosong" << endl;
            return;
        }

        cout << "Data antrian teller:" << endl;
        Node* current = front;
        int position = 1;
        while (current != nullptr) {
            cout << position << ". " << current->data << endl;
            current = current->next;
            position++;
        }
    }
};

int main() {
    Queue q;  // Membuat objek Queue

    // Menambahkan elemen ke antrian
    q.enqueue("Andi");
    q.enqueue("Maya");
    q.view();  // Menampilkan antrian
    cout << "Jumlah Antrian = " << q.count() << endl;

    // Menghapus elemen dari antrian
    q.dequeue();
    q.view();  // Menampilkan antrian setelah dequeue
    cout << "Jumlah Antrian = " << q.count() << endl;

    // Menghapus semua elemen dalam antrian
    q.clear();
    q.view();  // Menampilkan antrian setelah clear
    cout << "Jumlah Antrian = " << q.count() << endl;

    return 0;
}
```

Output

```
PS D:\Praktikum Struktur
n_8\Ungudied\" ; if ($?)
Data antrian teller:
1. Andi
2. Maya
Jumlah Antrian = 2
Data antrian teller:
1. Maya
Jumlah Antrian = 1
Antrian Kosong
Jumlah Antrian = 0
PS D:\Praktikum Struktur
```

2. Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

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

// Definisikan struktur Node untuk linked list
struct Node {
    string nama;  // Nama mahasiswa
    string nim;   // NIM mahasiswa
    Node* next;   // Pointer ke node berikutnya dalam list
};

// Definisikan kelas Queue untuk mengelola antrian berbasis linked list
class Queue {
private:
    Node* front;   // Pointer ke depan antrian
    Node* back;    // Pointer ke belakang antrian

public:
    // Konstruktor, inisialisasi queue dengan front dan back null
    Queue() : front(nullptr), back(nullptr) {}

    // Cek apakah antrian kosong
    bool isEmpty() {
        return front == nullptr;
    }

    // Menambahkan elemen ke dalam antrian (enqueue)
    void enqueue(const string& nama, const string& nim) {
        Node* newNode = new Node{nama, nim, nullptr};  // Buat node baru

        if (isEmpty()) {
            front = back = newNode;  // Jika kosong, front dan back menunjuk ke node baru
        } else {
            back->next = newNode;  // Sambungkan node baru di belakang antrian
            back = newNode;        // Update pointer back ke node baru
        }
    }
```

```
39.
40.     // Menghapus elemen dari antrian (dequeue)
41.     void dequeue() {
42.         if (isEmpty()) {
43.             cout << "Antrian Kosong" << endl;
44.             return;
45.         }
46.
47.         Node* temp = front;  // Simpan node depan sementara
48.         front = front->next; // Pindahkan front ke node berikutnya
49.         if (front == nullptr) {
50.             back = nullptr;  // Jika antrian kosong, back juga null
51.         }
52.         delete temp;  // Hapus node lama (node yang terdepan)
53.     }
54.
55.     // Menghitung jumlah elemen dalam antrian
56.     int count() {
57.         int count = 0;
58.         Node* current = front;
59.         while (current != nullptr) {
60.             count++;
61.             current = current->next;
62.         }
63.         return count;
64.     }
65.
66.     // Menghapus semua elemen dalam antrian
67.     void clear() {
68.         while (!isEmpty()) {
69.             dequeue();  // Dequeue hingga antrian kosong
70.         }
71.     }
72.
73.     // Menampilkan elemen-elemen dalam antrian
74.     void view() {
75.         if (isEmpty()) {
76.             cout << "Antrian Kosong" << endl;
77.             return;
78.         }
79.
80.         cout << "Data Antrian Mahasiswa:" << endl;
81.         Node* current = front;
82.         int position = 1;
83.         while (current != nullptr) {
84.             cout << position << ". Nama: " << current->nama << ", NIM: "
    << current->nim << endl;
85.             current = current->next;
86.             position++;
87.         }
88.     }
89. };
```

```
90.
91.int main() {
92.    Queue q;  // Membuat objek Queue
93.    int pilihan;
94.    string nama, nim;
95.
96.    do {
97.        cout << "\nMenu Antrian Mahasiswa:" << endl;
98.        cout << "1. Tambah Mahasiswa ke Antrian (Enqueue)" << endl;
99.        cout << "2. Hapus Mahasiswa dari Antrian (Dequeue)" << endl;
100.             cout << "3. Lihat Daftar Antrian Mahasiswa" << endl;
101.             cout << "4. Hapus Semua Antrian" << endl;
102.             cout << "5. Keluar" << endl;
103.             cout << "Pilih opsi: ";
104.             cin >> pilihan;
105.
106.             switch (pilihan) {
107.                 case 1:
108.                     cout << "Masukkan Nama Mahasiswa: ";
109.                     cin.ignore(); // Mengabaikan newline tersisa dari
   input sebelumnya
110.                     getline(cin, nama);
111.                     cout << "Masukkan NIM Mahasiswa: ";
112.                     getline(cin, nim);
113.                     q.enqueue(nama, nim);
114.                     cout << "Mahasiswa berhasil ditambahkan ke
   antrian." << endl;
115.                     break;
116.
117.                 case 2:
118.                     q.dequeue();
119.                     cout << "Mahasiswa di depan antrian telah
   dihapus." << endl;
120.                     break;
121.
122.                 case 3:
123.                     q.view();
124.                     cout << "Jumlah Antrian = " << q.count() << endl;
125.                     break;
126.
127.                 case 4:
128.                     q.clear();
129.                     cout << "Semua antrian telah dihapus." << endl;
130.                     break;
131.
132.                 case 5:
133.                     cout << "Keluar dari program." << endl;
134.                     break;
135.
136.                 default:
137.                     cout << "Pilihan tidak valid. Silakan coba lagi."
   << endl;
```

```
            }
        } while (pilihan != 5);

        return 0;
    }

```

Output

```
Menu Antrian Mahasiswa:
1. Tambah Mahasiswa ke Antrian (Enqueue)
2. Hapus Mahasiswa dari Antrian (Dequeue)
3. Lihat Daftar Antrian Mahasiswa
4. Hapus Semua Antrian
5. Keluar
Pilih opsi: 1
Masukkan Nama Mahasiswa: jidan
Masukkan NIM Mahasiswa: 232323
Mahasiswa berhasil ditambahkan ke antrian.

Menu Antrian Mahasiswa:
1. Tambah Mahasiswa ke Antrian (Enqueue)
2. Hapus Mahasiswa dari Antrian (Dequeue)
3. Lihat Daftar Antrian Mahasiswa
4. Hapus Semua Antrian
5. Keluar
Pilih opsi: 5
Keluar dari program.
PS D:\Praktikum Struktur Data\08_Pengenalan_CPP_Bagian_8\Ungudied>
```

3. Modifikasi program pada soal 1 sehingga mahasiswa dapat diprioritaskan berdasarkan NIM (NIM yang lebih kecil didahulukan pada saat output).

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

// Definisikan struktur Node untuk linked list
struct Node {
    string nama;  // Nama mahasiswa
    string nim;   // NIM mahasiswa
    Node* next;   // Pointer ke node berikutnya dalam list
};

// Definisikan kelas Queue untuk mengelola antrian berbasis linked list
class Queue {
private:
    Node* front;   // Pointer ke depan antrian
    Node* back;    // Pointer ke belakang antrian

public:
    // Konstruktor, inisialisasi queue dengan front dan back null
    Queue() : front(nullptr), back(nullptr) {}

    // Cek apakah antrian kosong
    bool isEmpty() {
        return front == nullptr;
```

```
    }

    // Menambahkan elemen ke dalam antrian (enqueue)
    void enqueue(const string& nama, const string& nim) {
        Node* newNode = new Node{nama, nim, nullptr};  // Buat node baru

        if (isEmpty()) {
            front = back = newNode;  // Jika kosong, front dan back menunjuk ke node baru
        } else {
            back->next = newNode;  // Sambungkan node baru di belakang antrian
            back = newNode;        // Update pointer back ke node baru
        }
    }

    // Menghapus elemen dari antrian (dequeue)
    void dequeue() {
        if (isEmpty()) {
            cout << "Antrian Kosong" << endl;
            return;
        }

        Node* temp = front;  // Simpan node depan sementara
        front = front->next; // Pindahkan front ke node berikutnya
        if (front == nullptr) {
            back = nullptr;  // Jika antrian kosong, back juga null
        }
        delete temp;  // Hapus node lama (node yang terdepan)
    }

    // Menghitung jumlah elemen dalam antrian
    int count() {
        int count = 0;
        Node* current = front;
        while (current != nullptr) {
            count++;
            current = current->next;
        }
        return count;
    }

    // Menghapus semua elemen dalam antrian
    void clear() {
        while (!isEmpty()) {
            dequeue();  // Dequeue hingga antrian kosong
        }
    }

    // Menampilkan elemen-elemen dalam antrian
    void view() {
        if (isEmpty()) {
```

```
            cout << "Antrian Kosong" << endl;
            return;
        }

        cout << "Data Antrian Mahasiswa:" << endl;
        Node* current = front;
        int position = 1;
        while (current != nullptr) {
            cout << position << ". Nama: " << current->nama << ", NIM: "
<< current->nim << endl;
            current = current->next;
            position++;
        }
    }
};

int main() {
    Queue q;  // Membuat objek Queue
    int pilihan;
    string nama, nim;

    do {
        cout << "\nMenu Antrian Mahasiswa:" << endl;
        cout << "1. Tambah Mahasiswa ke Antrian (Enqueue)" << endl;
              cout << "2. Hapus Mahasiswa dari Antrian (Dequeue)" <<
endl;
              cout << "3. Lihat Daftar Antrian Mahasiswa" << endl;
              cout << "4. Hapus Semua Antrian" << endl;
              cout << "5. Keluar" << endl;
              cout << "Pilih opsi: ";
              cin >> pilihan;

              switch (pilihan) {
                  case 1:
                      cout << "Masukkan Nama Mahasiswa: ";
                      cin.ignore(); // Mengabaikan newline tersisa dari
input sebelumnya
                      getline(cin, nama);
                      cout << "Masukkan NIM Mahasiswa: ";
                      getline(cin, nim);
                      q.enqueue(nama, nim);
                      cout << "Mahasiswa berhasil ditambahkan ke
antrian." << endl;
                      break;

                  case 2:
                      q.dequeue();
                      cout << "Mahasiswa di depan antrian telah
dihapus." << endl;
                      break;

                  case 3:
```

```
                    q.view();
                    cout << "Jumlah Antrian = " << q.count() << endl;
                    break;

                case 4:
                    q.clear();
                    cout << "Semua antrian telah dihapus." << endl;
                    break;

                case 5:
                    cout << "Keluar dari program." << endl;
                    break;

                default:
                    cout << "Pilihan tidak valid. Silakan coba lagi."
   << endl;
            }
        } while (pilihan != 5);

        return 0;
    }
```

Output

```
Menu Antrian Mahasiswa:
1. Tambah Mahasiswa ke Antrian (Enqueue)
2. Hapus Mahasiswa dari Antrian (Dequeue)
3. Lihat Daftar Antrian Mahasiswa
4. Hapus Semua Antrian
5. Keluar
Pilih opsi: 1
Masukkan Nama Mahasiswa: jidan
Masukkan NIM Mahasiswa: 232323
Mahasiswa berhasil ditambahkan ke antrian.

Menu Antrian Mahasiswa:
1. Tambah Mahasiswa ke Antrian (Enqueue)
2. Hapus Mahasiswa dari Antrian (Dequeue)
3. Lihat Daftar Antrian Mahasiswa
4. Hapus Semua Antrian
5. Keluar
Pilih opsi: 5
Keluar dari program.
PS D:\Praktikum Struktur Data\08_Pengenalan_CPP_Bagian_8\Ungudied>
```