**1. Header dan Namespace**

cpp

CopyEdit

#include <iostream>

using namespace std;

* **Fungsi:** Untuk menggunakan fungsi-fungsi standar seperti cout, cin, dll.

**2. Deklarasi dan Definisi Kelas Queues**

cpp

CopyEdit

class Queues {

int FRONT, REAR, max = 5;

int queue\_array[5];

* **Nama:** *Data members*
* **Fungsi:**
  + FRONT: Menyimpan indeks elemen depan antrian.
  + REAR: Menyimpan indeks elemen belakang antrian.
  + max: Ukuran maksimal antrian (5 elemen).
  + queue\_array[5]: Array untuk menyimpan elemen queue (antrian).

**3. Constructor**

cpp

CopyEdit

public:

Queues(){

FRONT = -1;

REAR = -1;

}

* **Nama:** *Constructor*
* **Fungsi:** Menginisialisasi antrian dalam keadaan kosong (FRONT dan REAR diset ke -1).

**4. Fungsi insert()**

cpp

CopyEdit

void insert() {

...

}

* **Nama:** *Insert Operation (enqueue)*
* **Fungsi:** Menambahkan elemen ke antrian dengan logika **circular queue**.

**Penjelasan detail:**

cpp

CopyEdit

if (FRONT == -1) { ... }

* Cek apakah antrian kosong. Jika iya, inisialisasi FRONT dan REAR ke 0.

cpp

CopyEdit

if (REAR == max - 1)

REAR = 0;

else

REAR = REAR + 1;

* Jika REAR sudah di ujung array (max - 1), maka balik ke awal (0). Ini yang membuatnya **circular**.

cpp

CopyEdit

queue\_array[REAR] = num;

* Simpan elemen ke array.

**5. Fungsi remove()**

cpp

CopyEdit

void remove() {

...

}

* **Nama:** *Delete Operation (dequeue)*
* **Fungsi:** Menghapus elemen dari depan antrian (FRONT).

**Penjelasan detail:**

cpp

CopyEdit

if (FRONT == -1) {

cout << "Queue underflow\n";

return;

}

* Cek apakah antrian kosong. Jika iya, tampilkan error.

cpp

CopyEdit

if (FRONT == REAR ){

FRONT = -1;

REAR = -1;

}

* Jika hanya 1 elemen tersisa, kosongkan antrian.

cpp

CopyEdit

if (FRONT == max - 1)

FRONT = 0;

else

FRONT = FRONT + 1;

* Untuk circular queue: jika FRONT di ujung, kembali ke awal.

**6. Fungsi display()**

cpp

CopyEdit

void display() {

...

}

* **Nama:** *Display Operation*
* **Fungsi:** Menampilkan seluruh elemen dalam antrian.

**Penjelasan detail:**

cpp

CopyEdit

if (FRONT == -1) {

cout << "Queue is empty\n";

return;

}

* Cek apakah antrian kosong.

cpp

CopyEdit

if (FRONT\_position <= REAR\_position) {

while (FRONT\_position <= REAR\_position) { ... }

}

* Jika FRONT lebih kecil dari REAR, iterasi dari FRONT ke REAR.

cpp

CopyEdit

else {

while (FRONT\_position <= max -1) { ... }

...

while (FRONT\_position <= REAR\_position) { ... }

}

* Jika FRONT > REAR, berarti data menyebar dari akhir ke awal (efek circular), sehingga tampilkan dua bagian array.

**7. Fungsi main()**

cpp

CopyEdit

int main() {

...

}

* **Nama:** *Driver Function / Program Utama*
* **Fungsi:** Menyediakan **menu interaktif** ke pengguna untuk mengakses insert, delete, display.

**Penjelasan:**

cpp

CopyEdit

while (true) {

...

switch (ch) {

case '1': q.insert(); break;

case '2': break;

case '3': break;

case '4': return 0;

* **Menu**:
  + 1: Panggil insert()
  + 2: [Harusnya] panggil remove() → **Saat ini kosong!**
  + 3: [Harusnya] panggil display() → **Saat ini kosong!**
  + 4: Keluar program

**Catatan**: Fungsi remove() dan display() **belum dipanggil di main()**, sehingga perlu ditambahkan:

cpp

CopyEdit

case '2':{

q.remove();

break;

}

case '3':{

q.display();

break;

}

**Kesimpulan Struktur Algoritma:**

| **Bagian** | **Nama Umum** | **Fungsi** |
| --- | --- | --- |
| Kelas Queues | Struktur Data | Menyimpan logika queue |
| insert() | Enqueue | Menambah elemen |
| remove() | Dequeue | Menghapus elemen |
| display() | Traversal | Menampilkan isi queue |
| main() | User Interface | Menu pilihan |

**💡 STRUKTUR UMUM PROGRAM**

cpp

CopyEdit

#include <iostream>

using namespace std;

**✅ Bagian ini:**

* Mengimpor library iostream agar bisa menggunakan cin, cout, dll.
* using namespace std; supaya tidak perlu menulis std::cout dan std::cin.

**🧱 DEKLARASI KELAS Queues**

cpp

CopyEdit

class Queues {

int FRONT, REAR, max = 5;

int queue\_array[5];

**✅ Penjelasan:**

* class Queues adalah struktur data berbasis **queue** (antrian).
* FRONT: menunjuk ke elemen **depan** antrian.
* REAR: menunjuk ke elemen **belakang** antrian.
* max: kapasitas maksimal antrian.
* queue\_array[5]: array untuk menyimpan data queue.

**🔧 CONSTRUCTOR: Inisialisasi Awal**

cpp

CopyEdit

Queues(){

FRONT = -1;

REAR = -1;

}

**✅ Penjelasan:**

* Antrian dianggap kosong saat FRONT dan REAR bernilai -1.
* Constructor ini dipanggil otomatis saat objek Queues dibuat di main.

**➕ FUNGSI insert(): Menambah data ke antrian**

cpp

CopyEdit

void insert() {

int num;

cout << "Enter a number: ";

cin >> num;

cout << endl;

**✅ Penjelasan:**

* Input angka dari user untuk dimasukkan ke queue.

cpp

CopyEdit

if (FRONT == -1) {

FRONT = 0;

REAR = 0;

}

**✅ Jika queue masih kosong:**

* Inisialisasi FRONT dan REAR ke 0, siap menambahkan elemen pertama.

cpp

CopyEdit

else {

if (REAR == max - 1)

REAR = 0;

else

REAR = REAR + 1;

}

**✅ Jika sudah ada data:**

* Jika REAR sudah di akhir array, mutar ke awal (REAR = 0) → **circular behavior**.
* Jika belum, lanjut ke indeks berikutnya.

cpp

CopyEdit

queue\_array[REAR] = num;

}

**✅ Menyimpan data:**

* Data dimasukkan ke indeks REAR.

**➖ FUNGSI remove(): Menghapus data dari antrian**

cpp

CopyEdit

void remove () {

if (FRONT == -1){

cout << "Queue underflow\n";

return;

}

**✅ Cek apakah antrian kosong:**

* Jika kosong (FRONT == -1), tidak bisa hapus data.

cpp

CopyEdit

cout << "\nThe element deleted from the queue is: " << queue\_array[FRONT] << "\n";

* Menampilkan data yang dihapus (elemen paling depan).

cpp

CopyEdit

if (FRONT == REAR ){

FRONT = -1;

REAR = -1;

}

**✅ Jika hanya ada 1 elemen:**

* Reset kembali queue ke kondisi kosong.

cpp

CopyEdit

else {

if (FRONT == max - 1)

FRONT = 0;

else

FRONT = FRONT + 1;

}

}

**✅ Jika lebih dari 1 elemen:**

* Jika FRONT sudah di ujung array, pindah ke indeks 0 (circular).
* Jika tidak, FRONT cukup ditambah satu.

**📄 FUNGSI display(): Menampilkan semua elemen dalam queue**

cpp

CopyEdit

void display(){

int FRONT\_position = FRONT;

int REAR\_position = REAR;

* Salin posisi FRONT dan REAR agar tidak mengubah aslinya.

cpp

CopyEdit

if (FRONT == -1) {

cout << "Queue is empty\n";

return;

}

* Cek apakah queue kosong.

cpp

CopyEdit

cout << "\nElement in the queue are...\n";

* Tampilkan isi queue.

**✅ Dua kondisi tampilan:**

**1. Jika queue tidak memutar**

cpp

CopyEdit

if (FRONT\_position <= REAR\_position){

while (FRONT\_position <= REAR\_position){

cout << queue\_array[FRONT\_position] << " ";

FRONT\_position++;

}

cout << endl;

}

**2. Jika queue memutar ke depan array**

cpp

CopyEdit

else {

while (FRONT\_position <= max -1) {

cout << queue\_array[FRONT\_position] << " ";

FRONT\_position++;

}

FRONT\_position = 0;

while (FRONT\_position <= REAR\_position){

cout << queue\_array[FRONT\_position] << " ";

FRONT\_position++;

}

cout << endl;

}

}

**🔁 PROGRAM UTAMA main()**

cpp

CopyEdit

int main(){

Queues q;

char ch;

* Membuat objek queue.
* Menyediakan input dari user (pakai switch).

cpp

CopyEdit

while (true){

try{

cout << "Menu" << endl;

cout << "1. Implemen insert operation" << endl;

cout << "2. Implemen delete operation" << endl;

cout << "3. Display values" << endl;

cout << "4. Exit" << endl;

cout << "Enter your choice (1-4): ";

cin >> ch;

cout << endl;

* Menampilkan menu ke user.

cpp

CopyEdit

switch (ch) {

case '1':{

q.insert();

break;

}

case '2':{

q.remove(); // Tambahkan ini!

break;

}

case '3':{

q.display(); // Tambahkan ini juga!

break;

}

case '4':{

return 0;

}

default: {

cout << "Invalid option!!" << endl;

break;

}

}

* Setiap pilihan memanggil fungsi sesuai kebutuhan.

cpp

CopyEdit

}

catch (exception& e) {

cout << "Check for the values entered." << endl;

}

}

return 0;

}