

LAPORAN PRAKTIKUM

PERTEMUAN 8

Queue



Nama :

Haza Zaidan Zidna Fann

(2311104056)

Dosen :

Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

Mahasiswa mampu menjelaskan definisi dan konsep dari queue

Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue

Mahasiswa mampu menerapkan operasi tampil data pada queue

II. LANDASAN TEORI

Queue adalah struktur data linier yang mengikuti prinsip First In, First Out (FIFO), di mana elemen pertama yang masuk akan menjadi elemen pertama yang keluar. Queue digunakan dalam berbagai aplikasi seperti antrian di loket, pengelolaan proses dalam sistem operasi, dan pengolahan data streaming.

Operasi dasar pada Queue:

Enqueue: Menambahkan elemen ke belakang (rear).

Dequeue: Menghapus elemen dari depan (front).

Peek/Front: Menampilkan elemen di depan tanpa menghapusnya.

isEmpty: Memeriksa apakah antrian kosong.

isFull: Memeriksa apakah antrian penuh.

Queue memiliki dua sisi utama:

Front: Elemen pertama yang akan keluar.

Rear: Elemen yang terakhir masuk.

Implementasi Queue: Queue dapat diimplementasikan menggunakan array atau linked list. Array lebih sederhana tetapi memerlukan pengelolaan ruang yang efisien, sementara linked list lebih fleksibel dalam ukuran.

III. GUIDED

```

1  #include <iostream>
2  #define MAX 100
3
4  using namespace std;
5
6  class Queue {
7  private:
8      int front, rear;
9      int arr[MAX];
10 public:
11
12     Queue() {
13         front = -1;
14         rear = -1;
15     }
16
17     bool isFull() {
18         return rear == MAX - 1;
19     }
20
21     bool isEmpty() {
22         return front == -1 || front > rear;
23     }
24
25     void enqueue(int x) {
26         if (isFull()) {
27             cout << "Queue Overflow\n";
28             return;
29         }
30         if (front == -1) front = 0;
31         arr[++rear] = x;
32     }
33
34     void dequeue() {
35         if (isEmpty()) {
36             cout << "Queue Underflow\n";
37             return;
38         }
39         front++;
40     }
41
42     int peek() {
43         if (!isEmpty()) {
44             return arr[front];
45         }
46         cout << "Queue is empty\n";
47         return -1;
48     }
49
50     void display() {
51         if (isEmpty()) {
52             cout << "Queue is empty\n";
53             return;
54         }
55         for (int i = front; i <= rear; i++) {
56             cout << arr[i] << " ";
57         }
58         cout << "\n";
59     }
60 };
61
62 int main() {
63     Queue q;
64
65     q.enqueue(10);
66     q.enqueue(20);
67     q.enqueue(30);
68
69     cout << "Queue elements: ";
70     q.display();
71
72     cout << "Front element: " << q.peek() << "\n";
73
74     cout << "After dequeuing, queue elements: ";
75     q.display();
76
77     return 0;
78 }

```

```
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 10 20 30
```

Deklarasi dan Konstruktor:

front dan rear: Mewakili indeks depan dan belakang queue. Kedua variabel ini diinisialisasi dengan nilai -1 yang menandakan bahwa queue kosong saat awal.

Konstruktor Queue(): Mengatur nilai awal front dan rear ke -1.

Metode isFull():

Memeriksa apakah queue sudah penuh dengan membandingkan rear dengan $MAX - 1$ (ukuran array - 1).

Metode isEmpty():

Memeriksa apakah queue kosong, yang terjadi jika front lebih besar dari rear atau keduanya -1.

Metode enqueue(x):

Menambahkan elemen x ke belakang antrian (dengan memindahkan rear).

Jika queue penuh, mencetak "Queue Overflow".

Metode dequeue():

Menghapus elemen depan antrian dengan meningkatkan nilai front.

Jika queue kosong, mencetak "Queue Underflow".

Metode peek():

Mengembalikan elemen di depan queue tanpa menghapusnya.

Jika queue kosong, mencetak "Queue is empty".

Metode display():

Menampilkan seluruh elemen dalam queue, dimulai dari front hingga rear.

Program Utama (main()):

Menambahkan 3 elemen (10, 20, 30) ke dalam queue menggunakan enqueue().

Menampilkan elemen-elemen dalam queue menggunakan display().

Menampilkan elemen depan menggunakan peek().

Menghapus elemen dari depan dengan dequeue() dan menampilkan elemen-elemen setelahnya.

```

1  #include <iostream>
2
3  using namespace std;
4
5  // Node untuk setiap elemen Queue
6  class Node {
7  public:
8      int data;        // Data elemen
9      Node* next;      // Pointer ke node berikutnya
10
11     // Konstruktor untuk Node
12     Node(int value) {
13         data = value;
14         next = nullptr;
15     }
16 };
17
18 // Kelas Queue menggunakan linked list
19 class Queue {
20 private:
21     Node* front; // Pointer ke elemen depan Queue
22     Node* rear;  // Pointer ke elemen belakang Queue
23
24 public:
25     // Konstruktor Queue
26     Queue() {
27         front = rear = nullptr;
28     }
29
30     // Mengecek apakah Queue kosong
31     bool isEmpty() {
32         return front == nullptr;
33     }
34
35     // Menambahkan elemen ke Queue
36     void enqueue(int x) {
37         Node* newNode = new Node(x);
38         if (isEmpty()) {
39             front = rear = newNode; // Jika Queue kosong
40             return;
41         }
42         rear->next = newNode; // Tambahkan node baru ke belakang
43         rear = newNode;      // Perbarui rear
44     }
45
46     // Menghapus elemen dari depan Queue
47     void dequeue() {
48         if (isEmpty()) {
49             cout << "Queue Underflow\n";
50             return;
51         }
52         Node* temp = front;    // Simpan node depan untuk dihapus
53         front = front->next;    // Pindahkan front ke node berikutnya
54         delete temp;           // Hapus node lama
55         if (front == nullptr)   // Jika Queue kosong, rear juga harus null
56             rear = nullptr;
57     }
58
59     // Mengembalikan elemen depan Queue tanpa menghapusnya
60     int peek() {
61         if (!isEmpty()) {
62             return front->data;
63         }
64         cout << "Queue is empty\n";
65         return -1; // Nilai sentinel
66     }
67
68     // Menampilkan semua elemen di Queue
69     void display() {
70         if (isEmpty()) {
71             cout << "Queue is empty\n";
72             return;
73         }
74         Node* current = front; // Mulai dari depan
75         while (current) {      // Iterasi sampai akhir
76             cout << current->data << " ";
77             current = current->next;
78         }
79         cout << "\n";
80     }
81 };
82
83 // Fungsi utama untuk menguji Queue
84 int main() {
85     Queue q;
86
87     // Menambahkan elemen ke Queue
88     q.enqueue(10);
89     q.enqueue(20);
90     q.enqueue(30);
91
92     // Menampilkan elemen di Queue
93     cout << "Queue elements: ";
94     q.display();
95
96     // Menampilkan elemen depan
97     cout << "Front element: " << q.peek() << "\n";
98
99     // Menghapus elemen dari depan Queue
100    q.dequeue();
101    cout << "After dequeuing, queue elements: ";
102    q.display();
103
104    return 0;
105 }

```

```
Queue elements: 10 20 30
Front element: 10
After dequeuing, queue elements: 20 30
```

Kelas Node:

data: Menyimpan nilai elemen.

next: Menunjuk ke node berikutnya dalam queue.

Konstruktor menginisialisasi data dengan nilai elemen dan next dengan nullptr.

Kelas Queue:

front: Pointer ke elemen depan queue.

rear: Pointer ke elemen belakang queue.

Fungsi:

isEmpty(): Mengecek apakah queue kosong (jika front adalah nullptr).

enqueue(x): Menambahkan elemen baru ke belakang queue.

Jika queue kosong, elemen menjadi front dan rear.

Jika tidak, elemen baru ditambahkan ke node setelah rear, lalu rear diperbarui.

dequeue(): Menghapus elemen depan dari queue.

Jika kosong, mencetak "Queue Underflow".

Jika elemen terakhir dihapus, front dan rear diatur ke nullptr.

peek(): Mengembalikan data dari elemen depan tanpa menghapusnya.

Jika kosong, mencetak "Queue is empty" dan mengembalikan -1.

display(): Menampilkan semua elemen queue dari front ke rear.

Fungsi main:

Menambahkan elemen ke dalam queue: enqueue(10), enqueue(20), enqueue(30).

Menampilkan elemen queue menggunakan display().

Menampilkan elemen depan menggunakan peek().

Menghapus elemen depan menggunakan dequeue(), lalu menampilkan elemen setelah penghapusan.

```

1  #include<iostream>
2
3  using namespace std;
4
5  const int maksimalQueue = 5; // Maksimal antrian
6  int front = 0; // Penanda antrian
7  int back = 0; // Penanda
8  string queueTeller[5]; // Fungsi pengecekan
9
10 bool isFull() { // Pengecekan antrian penuh atau tidak
11     if (back == maksimalQueue) { return true; // =1
12     } else {
13         return false;
14     }
15 }
16
17 bool isEmpty() { // Antriannya kosong atau tidak
18     if (back == 0) { return true;
19     } else {
20         return false;
21     }
22 }
23
24 void enqueueAntrian(string data) { // Fungsi menambahkan antrian
25     if (isFull()) {
26         cout << "Antrian penuh" << endl;
27     } else {
28         if (isEmpty()) { // Kondisi ketika queue kosong
29             queueTeller[0] = data; front++;
30             back++;
31         } else { // Antriannya ada isi queueTeller[back] = data; back++;
32         }
33     }
34 }
35
36 void dequeueAntrian() { // Fungsi mengurangi antrian
37     if (isEmpty()) {
38         cout << "Antrian kosong" << endl;
39     } else {
40         for (int i = 0; i < back; i++) { queueTeller[i] = queueTeller[i + 1];
41         }
42         back--;
43     }
44 }
45
46 int countQueue() { // Fungsi menghitung banyak antrian
47     return back;
48 }
49
50 void clearQueue() { // Fungsi menghapus semua antrian
51     if (isEmpty()) {
52         cout << "Antrian kosong" << endl;
53     } else {
54         for (int i = 0; i < back; i++) { queueTeller[i] = "";
55         }
56         back = 0;
57         front = 0;
58     }
59 }
60
61 void viewQueue() { // Fungsi melihat antrian
62     cout << "Data antrian teller:" << endl; for (int i = 0; i < maksimalQueue; i++) {
63         if (queueTeller[i] != "") {
64             cout << i + 1 << ". " << queueTeller[i] <<
65         }
66     } endl;
67
68     } else {
69         cout << i + 1 << ". (kosong)" << endl;
70     }
71 }
72 }
73 }
74 }
75
76 int main() {
77     enqueueAntrian("Andi");
78
79     enqueueAntrian("Maya");
80
81     viewQueue();
82     cout << "Jumlah antrian = " << countQueue() << endl;
83
84     dequeueAntrian();
85     viewQueue();
86     cout << "Jumlah antrian = " << countQueue() << endl;
87
88     clearQueue();
89     viewQueue();
90     cout << "Jumlah antrian = " << countQueue() << endl;
91
92     return 0;
93 }

```

```

Data antrian teller:
1. Andi
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
Antrian kosong
Data antrian teller:
1. (kosong)
2. (kosong)
3. (kosong)
4. (kosong)
5. (kosong)
Jumlah antrian = 0

```

Konstanta dan Variabel Global:

maksimalQueue: Batas maksimum queue (5 elemen).

front dan back: Penanda awal dan akhir queue.

queueTeller: Array untuk menyimpan elemen queue.

Fungsi Operasi Dasar:

isFull(): Memeriksa apakah queue penuh (jika back == maksimalQueue).

isEmpty(): Memeriksa apakah queue kosong (jika back == 0).

enqueueAntrian(string data): Menambahkan elemen ke belakang queue.

Jika penuh, mencetak "Antrian penuh".

Jika kosong, elemen pertama ditambahkan di indeks 0.

dequeueAntrian(): Menghapus elemen dari depan queue.

Jika kosong, mencetak "Antrian kosong".

Menggeser elemen setelah elemen pertama.

countQueue(): Menghitung jumlah elemen dalam queue (back menunjukkan jumlah elemen).

clearQueue(): Menghapus semua elemen dalam queue dengan mengatur front dan back ke 0.

viewQueue(): Menampilkan elemen queue dan posisi kosong.

Fungsi main():

Menambahkan elemen "Andi" dan "Maya" ke queue.

Menampilkan elemen dalam queue dan menghitung jumlah elemen.

Menghapus elemen depan (dequeue) dan menampilkan queue setelahnya.
Menghapus semua elemen queue dan menampilkan hasilnya.

IV. UNGUIDED

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  struct Node {
6      string data;
7      Node* next;
8  };
9
10 class Queue {
11 private:
12     Node* front;
13     Node* back;
14     int size;
15
16 public:
17     Queue() {
18         front = nullptr;
19         back = nullptr;
20         size = 0;
21     }
22
23     bool isFull() {
24         return false; // Linked list tidak memiliki batas ukuran tetap
25     }
26
27     bool isEmpty() {
28         return size == 0;
29     }
30
31     void enqueue(string data) {
32         Node* newNode = new Node();
33         newNode->data = data;
34         newNode->next = nullptr;
35
36         if (isEmpty()) {
37             front = newNode;
38             back = newNode;
39         } else {
40             back->next = newNode;
41             back = newNode;
42         }
43         size++;
44     }
45
46     void dequeue() {
47         if (isEmpty()) {
48             cout << "Antrian kosong" << endl;
49         } else {
50             Node* temp = front;
51             front = front->next;
52             delete temp;
53             size--;
54         }
55     }
56
57     int count() {
58         return size;
59     }
60
61     void clear() {
62         while (!isEmpty()) {
63             dequeue();
64         }
65     }
66
67     void view() {
68         cout << "Isi queue: ";
69         Node* current = front;
70         while (current != nullptr) {
71             cout << current->data << " ";
72             current = current->next;
73         }
74         cout << endl;
75     }
76 };
77
78 int main() {
79     Queue queue;
80
81     // Input data oleh user
82     int jumlahData;
83     cout << "Masukkan jumlah nama: ";
84     cin >> jumlahData;
85     cin.ignore(); // Membersihkan buffer newline
86
87     for (int i = 0; i < jumlahData; i++) {
88         string nama;
89         cout << "Masukkan nama ke-" << i + 1 << ": ";
90         getline(cin, nama);
91         queue.enqueue(nama);
92     }
93
94     cout << "\nData dalam queue setelah input:\n";
95     queue.view();
96
97     return 0;
98 }

```

```
Masukkan jumlah nama: 2
Masukkan nama ke-1: Haza
Masukkan nama ke-2: Zaidan

Data dalam queue setelah input:
Isi queue: Haza Zaidan
```

Node Structure:

Setiap elemen antrian diwakili oleh Node yang berisi data (nama) dan pointer next ke node berikutnya.

Queue Class:

Attributes:

front: Menunjuk elemen pertama dalam queue.

back: Menunjuk elemen terakhir dalam queue.

size: Menyimpan jumlah elemen dalam queue.

Methods:

enqueue: Menambahkan data ke akhir queue.

dequeue: Menghapus data dari depan queue.

view: Menampilkan semua data dalam queue.

clear: Menghapus semua elemen queue.

isEmpty: Mengecek apakah queue kosong.

Main Function:

Meminta pengguna memasukkan jumlah data dan elemen (nama) untuk dimasukkan ke dalam queue.

Menampilkan isi queue setelah input.

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  struct Mahasiswa {
6      string nama;
7      long long nim;
8  };
9
10 struct Node {
11     Mahasiswa data;
12     Node* next;
13 };
14
15 class Queue {
16 private:
17     Node* front;
18     Node* back;
19     int size;
20
21 public:
22     Queue() {
23         front = nullptr;
24         back = nullptr;
25         size = 0;
26     }
27
28     bool isFull() {
29         return false; // Linked list tidak memiliki batas ukuran tetap
30     }
31
32     bool isEmpty() {
33         return size == 0;
34     }
35
36     void enqueue(string nama, long long nim) {
37         Node* newNode = new Node();
38         newNode->data.nama = nama;
39         newNode->data.nim = nim;
40         newNode->next = nullptr;
41
42         if (isEmpty()) {
43             front = newNode;
44             back = newNode;
45         } else {
46             back->next = newNode;
47             back = newNode;
48         }
49         size++;
50     }
51
52     void dequeue() {
53         if (isEmpty()) {
54             cout << "Antrian kosong" << endl;
55         } else {
56             Node* temp = front;
57             front = front->next;
58             delete temp;
59             size--;
60         }
61     }
62
63     int count() {
64         return size;
65     }
66
67     void clear() {
68         while (!isEmpty()) {
69             dequeue();
70         }
71     }
72
73     void view() {
74         cout << "\nData mahasiswa dalam queue:\n" << endl;
75         Node* current = front;
76         int index = 1;
77         while (current != nullptr) {
78             cout << index << ". Nama: " << current->data.nama << ", NIM: " << current->data.nim << endl;
79             current = current->next;
80             index++;
81         }
82     }
83 };
84
85 int main() {
86     Queue queue;
87
88     // Input data mahasiswa oleh user
89     int jumlahMahasiswa;
90     cout << "Masukkan jumlah mahasiswa: ";
91     cin >> jumlahMahasiswa;
92     cin.ignore(); // Membersihkan buffer newline
93
94     for (int i = 0; i < jumlahMahasiswa; i++) {
95         string nama;
96         long long nim;
97         cout << "Masukkan Nama Mahasiswa: ";
98         getline(cin, nama);
99         cout << "Masukkan NIM Mahasiswa: ";
100        cin >> nim;
101        cin.ignore(); // Membersihkan buffer newline
102
103        queue.enqueue(nama, nim);
104    }
105
106    queue.view();
107
108    cout << "\nJumlah antrian = " << queue.count() << endl;
109    return 0;
110 }

```

```
Masukkan jumlah mahasiswa: 3
Masukkan Nama Mahasiswa: Haza
Masukkan NIM Mahasiswa: 2311104056
Masukkan Nama Mahasiswa: Zidna
Masukkan NIM Mahasiswa: 2311104020
Masukkan Nama Mahasiswa: Fann
Masukkan NIM Mahasiswa: 2311104079
```

```
Data mahasiswa dalam queue:
```

```
1. Nama: Haza, NIM: 2311104056
2. Nama: Zidna, NIM: 2311104020
3. Nama: Fann, NIM: 2311104079
```

```
Jumlah antrian = 3
```

Mahasiswa:

Struct untuk menyimpan data mahasiswa (nama dan NIM).

Node:

Struct untuk menyusun linked list, setiap node berisi Mahasiswa dan pointer ke node berikutnya (next).

Queue Class

Atribut:

front: Pointer ke elemen pertama (kepala) dalam queue.

back: Pointer ke elemen terakhir (ekor) dalam queue.

size: Menyimpan jumlah elemen dalam queue.

Method Utama:

enqueue: Menambahkan data mahasiswa di akhir queue.

dequeue: Menghapus data mahasiswa dari awal queue.

view: Menampilkan semua data mahasiswa dalam queue.

count: Mengembalikan jumlah elemen dalam queue.

clear: Menghapus semua elemen dari queue.

isEmpty: Mengecek apakah queue kosong.

Main Program

Meminta pengguna untuk memasukkan jumlah mahasiswa.

Menginput nama dan NIM mahasiswa, lalu menambahkannya ke dalam queue menggunakan enqueue.

Menampilkan seluruh data mahasiswa menggunakan view.

Menampilkan jumlah elemen dalam queue dengan count.

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  struct Mahasiswa {
6      string nama;
7      long long nim;
8  };
9
10 struct Node {
11     Mahasiswa data;
12     Node* next;
13 };
14
15 class Queue {
16 private:
17     Node* front;
18     Node* back;
19     int size;
20
21 public:
22     Queue() {
23         front = nullptr;
24         back = nullptr;
25         size = 0;
26     }
27
28     bool isFull() {
29         return false; // Linked list tidak memiliki batas ukuran tetap
30     }
31
32     bool isEmpty() {
33         return size == 0;
34     }
35
36     void enqueue(string nama, long long nim) {
37         Node* newNode = new Node();
38         newNode->data.nama = nama;
39         newNode->data.nim = nim;
40         newNode->next = nullptr;
41
42         if (isEmpty()) {
43             front = newNode;
44             back = newNode;
45         } else {
46             if (newNode->data.nim < front->data.nim) {
47                 newNode->next = front;
48                 front = newNode;
49             } else {
50                 Node* current = front;
51                 while (current->next != nullptr && current->next->data.nim < newNode->data.nim) {
52                     current = current->next;
53                 }
54                 newNode->next = current->next;
55                 current->next = newNode;
56                 if (newNode->next == nullptr) {
57                     back = newNode;
58                 }
59             }
60         }
61         size++;
62     }
63
64     void dequeue() {
65         if (isEmpty()) {
66             cout << "Antrian kosong" << endl;
67         } else {
68             Node* temp = front;
69             front = front->next;
70             delete temp;
71             size--;
72         }
73     }
74
75     int count() {
76         return size;
77     }
78
79     void clear() {
80         while (!isEmpty()) {
81             dequeue();
82         }
83     }
84
85     void view() {
86         cout << "Data mahasiswa dalam queue:" << endl;
87         Node* current = front;
88         int index = 1;
89         while (current != nullptr) {
90             cout << index << ". Nama: " << current->data.nama << ", NIM: " << current->data.nim << endl;
91             current = current->next;
92             index++;
93         }
94     }
95 };
96
97 int main() {
98     Queue queue;
99
100    // Input data mahasiswa oleh user
101    int jumlahMahasiswa;
102    cout << "Masukkan jumlah mahasiswa: ";
103    cin >> jumlahMahasiswa;
104    cin.ignore(); // Membersihkan buffer newline
105
106    for (int i = 0; i < jumlahMahasiswa; i++) {
107        string nama;
108        long long nim;
109        cout << "Masukkan Nama Mahasiswa: ";
110        getline(cin, nama);
111        cout << "Masukkan NIM Mahasiswa: ";
112        cin >> nim;
113        cin.ignore(); // Membersihkan buffer newline
114        queue.enqueue(nama, nim);
115    }
116
117    cout << "\nData mahasiswa setelah diurutkan berdasarkan NIM:\n";
118    queue.view();
119
120    cout << "\nJumlah antrian = " << queue.count() << endl;
121    return 0;
122 }
123

```

```
Masukkan jumlah mahasiswa: 2
Masukkan Nama Mahasiswa: Haza Zaidan
Masukkan NIM Mahasiswa: 2311104056
Masukkan Nama Mahasiswa: Zidna Fann
Masukkan NIM Mahasiswa: 2311104080

Data mahasiswa setelah diurutkan berdasarkan NIM:
Data mahasiswa dalam queue:
1. Nama: Haza Zaidan, NIM: 2311104056
2. Nama: Zidna Fann, NIM: 2311104080
```

Mahasiswa:

Struct untuk menyimpan informasi mahasiswa (nama dan nim).

Node:

Struct untuk membentuk linked list, berisi data (bertipe Mahasiswa) dan pointer next ke node berikutnya.

Queue Class

Atribut:

front: Pointer ke elemen pertama dalam queue.

back: Pointer ke elemen terakhir dalam queue.

size: Menyimpan jumlah elemen dalam queue.

Method Utama:

enqueue:

Menambahkan elemen ke queue berdasarkan prioritas NIM (terkecil di depan).

Jika queue kosong, elemen langsung menjadi front dan back.

Jika NIM elemen baru lebih kecil dari front, elemen baru menjadi front.

Jika NIM elemen baru lebih besar, dicari posisi yang sesuai untuk menyisipkan elemen.

dequeue:

Menghapus elemen dari depan queue (front).

view:

Menampilkan elemen queue dalam urutan dari front ke back.

count:

Mengembalikan jumlah elemen dalam queue.

V. KESIMPULAN

Queue adalah struktur data berbasis FIFO (First In, First Out), di mana elemen pertama yang masuk akan menjadi elemen pertama yang keluar, seperti antrian dalam kehidupan sehari-hari. Operasi utama meliputi enqueue (menambahkan elemen di akhir antrian) dan dequeue (menghapus elemen dari awal antrian). Queue dapat diimplementasikan menggunakan array atau linked list, dan sering digunakan dalam sistem komputasi seperti manajemen proses, antrian printer, atau pemrosesan data real-time. Variasi seperti priority queue memungkinkan elemen diproses berdasarkan prioritas, bukan hanya urutan masuk.

