PRAKTIKUM STRUKTUR DATA

MODUL 5 DAN 6

“STACK (TUMPUKAN) dan QUEUE”

Disusun oleh :

Diana Eka Permata Sari

TRPL 2C

(362458302090)

PROGAM STUDI TEKNOLOGI REKAYASA PERANKAT LUNAK
JURUSAN BISNIS DAN INFORMATIKA

2025

Jalan Raya Jember No.KM13, Kawang, Labanasem, Kec. Kabat,
Kabupaten Banyuwangi, Jawa Timur 68461

Modul 5 STACK (Tumpukan)

Tujuan :

1. Memahami konsep penyimpanan data dengan stack (tumpukan)
2. Memahami operasi pada stack
3. Mampu mengimplementasikan struktur data stack pada pemrograman berbasis obyek.

Dasar Teori :

Salah satu konsep yang efektif untuk menyimpan dan mengambil data adalah "terakhir masuk sebagai pertama yang keluar" (Last In First Out/LIFO). Dengan konsep ini, pengambilan data akan berkebalikan urutannya dengan penyimpanan data. Stack (tumpukan) adalah sebuah kumpulan data dimana data yang diletakkan di atas data yang lain. Dengan demikian stack adalah struktur data yang menggunakan konsep LIFO. Elemen terakhir yang disimpan dalam stack menjadi elemen pertama yang diambil. Dalam proses komputasi, untuk meletakkan sebuah elemen pada bagian atas stack disebut dengan push. Dan untuk memindahkan dari tempat teratas tersebut, kita melakukan pop.

Ada 2 operasi paling dasar dari stack yang dapat dilakukan, yaitu :

1. Operasi push yaitu operasi menambahkan elemen pada urutan terakhir (paling atas)
2. Operasi pop yaitu operasi mengambil sebuah elemen data pada urutan terakhir dan menghapus elemen tersebut dari stack.

1. Dart Stack Collection

   Package Dart juga menyediakan class Stack pada Stack, yang merupakan subclass dari Vector yang menggunakan standar last-in first-out (LIFO). Untuk menggunakan package ini, perlu melakukan instalasi menggunakan perintah pemasangan packade pada dev_dependencies. Untuk menjalankan Class Stack, sebelumnya anda perlu menuliskan perintah import 'package:stack/stack.dart'. Class Stack hanya digunakan untuk menentukan default constructor, untuk membuat stack kosong.
2. Mengubah Notasi Postfix menjadi Infix dengan Stack
   Salah satu penggunaan stack adalah mengubah notasi infix menjadi postfix. Berikut ini adalah algoritma untuk mengubah notasi infix menjadi notasi postfix:
   a. Baca ungkapan dalam notasi infix, misalnya S, tentukan panjang ungkapan tersebut, misalnya N karakter, siapkan sebuah stack kosong dan siapkan derajad masing- masing operator, misalnya: ^ berderajad 3, * dan / berderajad 2, + dan – berderajad 1 dan (berderajad 0).
   b. Dimulai dari i = 1 sampai N kerjakan langkah-langkah sebagai berikut:
      - R = S[I]
      - Test nilai R. Jika R adalah: Operand : Langsung ditulis Kurung buka : Push ke dalam tumpukan kurung tutup : Pop dan tulis semua isi tumpukan sampai ujung tumpukan = '('. Pop juga tanda '(' ini, tetapi tidak usah ditulis

Operator : Jika tumpukan kosong atau derajad R lebih tinggi dibanding derajad ujung tumpukan, push operator ke dalam tumpukan. Jika tidak, pop ujung tumpukan dan tulis; kemudian ulangi pembandingan R dengan ujung tumpukan. Kenudian R di-push

c. Jika akhir notasi infix telah tercapai, dan tumpukan masih belum kosong, pop semua isi tumpukan dan tulis hasilnya Untuk memahami algoritma di atas, kita coba mengubah ungkapan berikut, yang ditulis menggunakan notasi infix, menjadi notasi postfix (A + B) / ((C – D) * E ^ F)

Tugas Pendahuluan :

1. Jelaskan pengertian tentang Stack
   Jawab: Stak yaitu data yang mengikuti prinsip last in, first out (LIFO), Dimana elemen terakhir yang dimasukkan ke dalam stack adalah elemen pertama yang dikeluarkan. Didalam stack ada 2 operasi utama yaitu :
   - **Push**: Menambahkan elemen ke bagian atas stack.
   - **Pop**: Menghapus dan mengembalikan elemen teratas dari stack.
2. Jelaskan operasi-operasi pada stack yaitu :
   a. empty()
   b. top()
   c. pop()
   d. push(‘T element’)

   jawab:

   a. empty()
      - Deskripsi: Operasi ini digunakan untuk mengecek apakah stack kosong.
      - Return Value: Mengembalikan `true` jika stack tidak memiliki elemen, dan `false` jika stack memiliki satu atau lebih elemen.

   b. Top()
      - Deskripsi: Operasi ini mengembalikan elemen teratas dari stack tanpa menghapusnya.
      - Return Value: Mengembalikan nilai elemen yang berada di puncak stack. Jika stack kosong, biasanya akan menghasilkan kesalahan atau exception.

   c. pop()
      - Deskripsi: Operasi ini digunakan untuk menghapus elemen teratas dari stack.
      - Return Value: Mengembalikan nilai elemen yang dihapus dari puncak stack. Jika stack kosong, biasanya akan menghasilkan kesalahan atau exception

   d. push(‘T element’)
      - Deskripsi: Operasi ini digunakan untuk menambahkan elemen baru ke puncak stack.
      - Parameter: 'T element' adalah elemen yang ingin ditambahkan ke stack.
      - Return Value: Tidak ada (void), tetapi stack akan bertambah satu elemen dengan elemen baru di puncaknya.
3. Ubahlah ekspresi infix dibawah ini menjadi ekspresi postfix.

a. (A * B) + (C -D)
b. (A - B) - (C * D) +E
c. A* (B - C) - (D * E)

Jawab :

a. (A * B) + (C -D)
   Postfix : A B * C D - +
b. (A - B) - (C * D) +E
   Postfix : A B – C D * - E +
c. A* (B - C) - (D * E)
   Postfix : A B C - * D E * -

Percobaan :

1. Implementasi Stack dengan List, namun menggunakan perintah native dart

```
class Stack {
  int last_elemen = 0;
  int maxStack = 0;
  List<int> elemen = [];

  void StackOperation(int maxElements) {
    last_elemen = -1;
    maxStack = maxElements - 1;
    elemen = List<int>.filled(maxElements, 0);
  }

  bool isEmpty() {
    return last_elemen == -1;
  }

  bool isFull() {
    return last_elemen == maxStack;
  }

  void push(int data) {
    if (isFull()) {
      print("stack overflow");
    } else {
      last_elemen = last_elemen + 1;
      elemen[last_elemen] = data;
    }
  }

  int pop() {
    int data = 0;
    if (isEmpty()) {
      print("stack Underflow");
    } else {
```

```
      data = elemen[last_elemen];
      elemen[last_elemen] = 0;
      last_elemen = last_elemen - 1;
    }
    return data;
  }

  int top() {
   if (isEmpty()) {
    print("Stack Underflow");
    return 0;
  } else {
    return elemen[last_elemen];
    }
  }

  void printStack() {
    if (!isEmpty()) {
      print("Menampilkan urutan dari posisi tertinggi");
      for (int i = last_elemen; i > -1; i--) {
        print("Elemen ke-" + i.toString() + " = " +
   elemen[i].toString());
      }
    } else {
      print("Stack masih kosong");
    }
  }
}

void main() {
   Stack s = Stack();
  s.StackOperation(100);
  s.push(10);
  s.push(20);
  s.push(30);

  s.pop();
  print("Last Element of Stack: " + s.top().toString());
  s.printStack();
}
```

Hasil Run :

```
Last Element of Stack: 20
Menampilkan urutan dari posisi tertinggi
Elemen ke-1 = 20
Elemen ke-0 = 10

Exited.
```

2. Implementasi Stack dengan List Menggunakan Stack Class

```dart
class Stack<T> {
  final List<T> _stack = [];

  void push(T element) {
    _stack.add(element);
  }

  T pop() {
    if (_stack.isEmpty) {
      throw StateError("No elements in the Stack");
    } else {

  T lastElement = _stack.last;
    _stack.removeLast();
      return lastElement;
    }
  }

  T top() {
    if (_stack.isEmpty) {
      throw StateError("No elements in the Stack");
    } else {
      return _stack.last;
    }
  }

  bool isEmpty() {
    return _stack.isEmpty;
  }

  @override
  String toString() => _stack.toString();
}

void main() {
  var stack = Stack<int>();
  var test = stack.isEmpty();
  print(test); // Output: true
  stack.push(1);
  stack.push(2);
```

```
43.   stack.push(3);
44.   stack.push(4);
45.   stack.push(5);
46.   stack.push(7);
47.   print(stack); // Output: [1, 2, 3, 4, 5, 7]
48.
49.   var myNumber = stack.pop();
50.   print('My 1st number is $myNumber');
51.   var mySecNumber = stack.pop();
52.   print('My 2nd number is $mySecNumber');
53.
54.   print(stack.pop()); // Output: 4
55.   print(stack); // Output: [1, 2, 3]
56.
57.   print(stack.top()); // Output: 3
58.   print(stack.isEmpty()); // Output: false
59. }
```

Hasil Run :

```
true
[1, 2, 3, 4, 5, 7]
My 1st number is 7
My 2nd number is 5
4
[1, 2, 3]
3
false

Exited.
```

3. Implementasi Stack dengan package Stack Dart

```dart
import 'package:stack/stack.dart';
Run | Debug
void main() {
  // Membuat objek tumpukan
  var stack = Stack<int>();
  // Menambahkan beberapa elemen ke dalam tumpukan
  stack.push(1);
  stack.push(2);
  stack.push(3);
  // Mencetak isi tumpukan
  print('Stack: $stack');
  // Menghapus dan mencetak elemen teratas dari tumpukan
  print('Popped element: ${stack.pop()}');
  print('Stack: $stack');
  import 'package:stack/stack.dart'
void main() {
  // Membuat objek tumpukan
  var stack = Stack<int>();
  // Menambahkan beberapa elemen ke dalam tumpukan
  stack.push(1);
  stack.push(2);
  stack.push(3);
  // Mencetak isi tumpukan
  print('Stack: $stack');
  // Menghapus dan mencetak elemen teratas dari tumpukan
  print('Popped element: ${stack.pop()}');
  print('Stack: $stack');
```

4. Mengubah notasi infix menjadi postfix

```dart
import 'dart:io';

class Stack {
  List<String> _items = [];

  void push(String item) {
    _items.add(item);
  }

  String pop() {
    if (isEmpty()) {
      throw Exception('Stack is empty');
    }
    return _items.removeLast();
  }

  String top() {
    if (isEmpty()) {
      throw Exception('Stack is empty');
    }
    return _items.last;
  }
```

```
    bool isEmpty() {
      return _items.isEmpty;
    }
  }

  class PostfixToInfix {
    String postfixExp = "";

    String toInfix(String postfixExp) {
      var stack = Stack();
      for (var ch in postfixExp.split('')) {
        if (_isOperator(ch)) {
          var operand2 = stack.pop();
          var operand1 = stack.pop();
          var result = '($operand1$ch$operand2)';
          stack.push(result);
        } else {
          stack.push(ch);
        }
      }
      return stack.pop();
    }

    bool _isOperator(String ch) {
      return ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch ==
  '^';
    }
  }

  void main() {
    var pti = PostfixToInfix();
    String postfixExp;

    stdout.write("Postfix Expression : ");
    postfixExp = stdin.readLineSync()!;
    print("Infix Expression : ${pti.toInfix(postfixExp)}");
  }
```

Hasil Run :

```
PS D:\dartsdk-windows-x64-release\sd\dart_application_1\bin> dart run percoban4_modul5.dart
Postfix Expression : 52
Infix Expression : 2
PS D:\dartsdk-windows-x64-release\sd\dart_application_1\bin>
```

Latihan :

1. Buatlah program/class untuk:
   a. Konversi dari nilai desimal ke nilai biner, oktal dan heksadesimal. Contoh : Masukkan nilai desimal = 25 Hasil nilai biner = 11001 Hasil nilai oktal = 31 Hasil nilai heksadesimal = 19
   b. Membalik kalimat dan menentukan sebuah kalimat termasuk palindrom atau bukan Masukkan kalimat : algoritma dan struktur data Hasil = atad rutkurts nad amtirogla Bukan palindrom Masukkan kalimat : sugus Hasil = sugus Palindrom

```
2. import 'dart:io';
3.
4. class Converter {
5.   // Fungsi untuk mengkonversi desimal ke biner, oktal, dan
   heksadesimal
6.   void convertDecimal(int decimal) {
7.     String binary = decimal.toRadixString(2);
8.     String octal = decimal.toRadixString(8);
9.     String hexadecimal = decimal.toRadixString(16).toUpperCase();
10.
11.    print('Hasil nilai biner = $binary');
12.    print('Hasil nilai oktal = $octal');
13.    print('Hasil nilai heksadesimal = $hexadecimal');
14.  }
15.}
16.
17.class PalindromeChecker {
18.  // Fungsi untuk membalik kalimat dan memeriksa palindrom
19.  void checkPalindrome(String sentence) {
20.    String reversed = sentence.split('').reversed.join('');
21.    bool isPalindrome = sentence == reversed;
22.
23.    print('Hasil = $reversed');
24.    print(isPalindrome ? 'Palindrom' : 'Bukan palindrom');
25.  }
26.}
27.
28.void main() {
29.  // Bagian konversi desimal
30.  print('Masukkan nilai desimal: ');
31.  int decimal = int.parse(stdin.readLineSync()!);
32.  Converter converter = Converter();
33.  converter.convertDecimal(decimal);
34.
35.  // Bagian pemeriksaan palindrom
36.  print('Masukkan kalimat: ');
37.  String sentence = stdin.readLineSync()!;
38.  PalindromeChecker checker = PalindromeChecker();
39.  checker.checkPalindrome(sentence);
```

```
40. }
41.
```

Penjelasan :

1. **Converter Class**:
   - Mengandung metode convertDecimal yang menerima nilai desimal dan mengkonversinya menjadi biner, oktal, dan heksadesimal.
2. **PalindromeChecker Class**:
   - Mengandung metode checkPalindrome yang membalik kalimat dan memeriksa apakah kalimat tersebut palindrom.
3. **Main Function**:
   - Menerima input dari pengguna untuk nilai desimal dan kalimat, kemudian menggunakan kedua kelas untuk melakukan konversi dan pemeriksaan palindrom.

Hasil runnya :

```
PS D:\dartsdk-windows-x64-release\sd\dart_application_1\bin> dart run latihan1_modul5.dart
Masukkan nilai desimal:
25
Hasil nilai biner = 11001
Hasil nilai oktal = 31
Hasil nilai heksadesimal = 19
Masukkan kalimat:
sugus
Hasil = sugus
Palindrom
PS D:\dartsdk-windows-x64-release\sd\dart_application_1\bin>
```

2. Dari soal latihan 1 untuk soal yang sama tetapi menggunakan stack dengan list seperti pada percobaan 1 dan 2!

```dart
1. import 'dart:io';
2.
3. class Stack<T> {
4.   List<T> _elements = [];
5.
6.   void push(T element) {
7.     _elements.add(element);
8.   }
9.
10.  T pop() {
11.    if (_elements.isEmpty) {
12.      throw Exception('Stack is empty');
13.    }
```

```
    return _elements.removeLast();
  }

  bool isEmpty() {
    return _elements.isEmpty;
  }

  void clear() {
    _elements.clear();
  }
}

class Converter {
  // Fungsi untuk mengkonversi desimal ke biner, oktal, dan heksadesimal menggunakan stack
  void convertDecimal(int decimal) {
    Stack<int> stack = Stack<int>();
    int tempDecimal = decimal;

    // Konversi ke biner
    while (tempDecimal > 0) {
      stack.push(tempDecimal % 2);
      tempDecimal ~/= 2;
    }

    String binary = '';
    while (!stack.isEmpty()) {
      binary += stack.pop().toString();
    }

    // Reset dan konversi ke oktal
    tempDecimal = decimal;
    while (tempDecimal > 0) {
      stack.push(tempDecimal % 8);
      tempDecimal ~/= 8;
    }

    String octal = '';
    while (!stack.isEmpty()) {
      octal += stack.pop().toString();
    }

    // Reset dan konversi ke heksadesimal
    tempDecimal = decimal;
    while (tempDecimal > 0) {
      stack.push(tempDecimal % 16);
      tempDecimal ~/= 16;
    }
```

```
61.
62.     String hexadecimal = '';
63.     while (!stack.isEmpty()) {
64.       int hexValue = stack.pop();
65.       hexadecimal += hexValue < 10 ? hexValue.toString() :
   String.fromCharCode(hexValue - 10 + 'A'.codeUnitAt(0));
66.     }
67.
68.     print('Hasil nilai biner = $binary');
69.     print('Hasil nilai oktal = $octal');
70.     print('Hasil nilai heksadesimal = $hexadecimal');
71.   }
72.}
73.
74.class PalindromeChecker {
75.   // Fungsi untuk membalik kalimat menggunakan stack dan memeriksa
   palindrom
76.   void checkPalindrome(String sentence) {
77.     Stack<String> stack = Stack<String>();
78.
79.     // Push setiap karakter ke stack
80.     for (var char in sentence.split('')) {
81.       stack.push(char);
82.     }
83.
84.     String reversed = '';
85.     while (!stack.isEmpty()) {
86.       reversed += stack.pop();
87.     }
88.
89.     bool isPalindrome = sentence == reversed;
90.
91.     print('Hasil = $reversed');
92.     print(isPalindrome ? 'Palindrom' : 'Bukan palindrom');
93.   }
94.}
95.
96.void main() {
97.   // Bagian konversi desimal
98.   print('Masukkan nilai desimal: ');
99.   int decimal = int.parse(stdin.readLineSync()!);
100.        Converter converter = Converter();
101.        converter.convertDecimal(decimal);
102.
103.        // Bagian pemeriksaan palindrom
104.        print('Masukkan kalimat: ');
105.        String sentence = stdin.readLineSync()!;
106.        PalindromeChecker checker = PalindromeChecker();
```

```
107.         checker.checkPalindrome(sentence);
108.       }
109.
```

Penjelasan :

1. **Stack Class**:
   - Mewakili struktur data stack dengan metode push, pop, dan isEmpty untuk mengelola elemen.
2. **Converter Class**:
   - Menggunakan stack untuk mengkonversi angka desimal ke biner, oktal, dan heksadesimal.
3. **PalindromeChecker Class**:
   - Menggunakan stack untuk membalik kalimat dan memeriksa apakah kalimat tersebut palindrom.
4. **Main Function**:
   - Menerima input dari pengguna dan menggunakan kedua kelas untuk melakukan konversi dan pemeriksaan palindrom.

Hasil Run :

```
PS D:\dartsdk-windows-x64-release\sd\dart_application_1\bin> dart run latihan2_modul5.dart
Masukkan nilai desimal:
25
Hasil nilai biner = 11001
Hasil nilai oktal = 31
Hasil nilai heksadesimal = 19
Masukkan kalimat:
sugus
Hasil = sugus
Palindrom
PS D:\dartsdk-windows-x64-release\sd\dart_application_1\bin>
```

Modul 6 QUEUE

Tujuan :

1. Memahami konsep dan implementasi dari Queue
2. Memahami operasi-operasi pada Queue

Dasar Teori :

Antrian (Queue) dapat diartikan sebagai suatu kumpulan data yang seolah-olah terlihat seperti ada data yang ditelakkan di sebelah data yang lain seperti pada gambar 1. Pada gambar, data masuk melalui lorong di sebelah kanan dan masuk dari terowongan sebelah kiri. Hal ini

membuat antrian bersifat FIFO (First In First Out), beda dengan stack yang berciri LIFO. Antrian dapat dibuat baik dengan Array maupun dengan struct. Pada pembuatan antrian dengan Array, antrian yang disajikan bersifat statis. Ini disebabkan oleh jumlah maksimal Array sudah ditentukan sejak deklarasi awal. Ada 4 operasi dasar yang dapat dilakukan pada struktur data antrian, yakni:

a. Create (Antrian)
b. isEmpty (Antrian)
c. Insert (Elemen Antrian)
d. Remove (Antrian)

Algoritma

Terdapat satu buah pintu masuk di suatu ujung dan satu buah pintu keluar di ujung satunya, sehingga membutuhkan variabel Front dan Rear.

Create

Untuk menciptakan dan menginisialisasi Queue, dengan cara membuat Front dan Rear = -1

IsEmpty()

Untuk memeriksa apakah antrian suda penuh atau belum adalah dengan cara memerksa nilai Rear, jika Rear = -1 maka empty. Kita tidak memeriksa Front, karena Front adalah tanda untuk kepala Antrian (elemen pertama dalam antrian) yang tidak akan berubah-ubah. Pergerakan pada antrian terjadi dengan penambahan elemen antrian kebelakang, yaitu menggunakan nilai Rear.

IsFull()

Untuk mengecek apakah antrian sudah penuh atau belum adalah dengan cara mengecek nilai Rear, jika Rear >= MAX-1 (karena MAX-1 adalah batas elemen array pada C). Ini berarti sudah penuh.

Enqueue(data)

Untuk menambahkan elemen ke dalam antrian. Penambahan elemen selalu ditambahkan di elemen paling belakang. Penambahan elemen selalu menggerakkan variabel Rear dengan cara increment counter Rear.

Dequeue()

Digunakan untuk menghapus elemen terdepan/pertama dari Antrian. Dengan cara mengurangi Counter Rear dan menggeser semua elemen antria ke depan. Penggeseran dilakukan dengan menggunakan looping.

Clear()

Untuk menghapus elemen-elemen Antrian dengan cara membuat Rear dan Front = -1. Penghapusan elemen-elemen Antrian sebenarnya tidak menghapus arraynya, namun hanya mengeset indeks pengaksesannya ke nilai -1 sehingga elemen-elemen antrian tidak lagi terbaca.

Circular Bounded Queue

Cara mensimulasikan antrian secara Circular dalam Array Linear menggunakan arithmetic modular. arithmetic modular menggunakan ekspresi rumus (X % N) untuk menjaga besarnya nilai X pada range 0:N-1. Jika indeks telah sampai pada N dengan penambahan atau pengurangan tersebut, maka indeks akan diset pada angka 0.

Tugas Pendahuluan :

1. Buatlah resume 1 halaman mengenai Queue dan berikan penjelasannya!
   Jawab :
   Queue adalah struktur data linear yang mengikuti prinsip FIFO (First In, First Out), di mana elemen yang pertama kali dimasukkan adalah yang pertama kali dikeluarkan. Queue banyak digunakan dalam berbagai aplikasi, seperti penjadwalan tugas, pengolahan data, dan komunikasi antara proses.

   Operasi Dasar Queue
   1. **Enqueue**: Menambahkan elemen baru ke bagian belakang queue.
   2. **Dequeue**: Menghapus elemen dari bagian depan queue.
   3. **Peek/Front**: Melihat elemen terdepan tanpa menghapusnya.
   4. **IsEmpty**: Memeriksa apakah queue kosong.
   5. **Size**: Menghitung jumlah elemen dalam queue.

   Queue dapat diimplementasikan menggunakan:

   a. **Array**: Memiliki ukuran tetap, dan memerlukan penanganan saat mencapai batas.
   b. **Linked List**: Lebih fleksibel dalam ukuran, tetapi memerlukan lebih banyak memori untuk menyimpan pointer.

   Kelebihannya:

   1. Sederhana dan mudah dipahami.
   2. Efisien dalam penggunaan memori (terutama dengan linked list).

   Kekurangannya:

   1. Implementasi menggunakan array memiliki batasan ukuran.
   2. Operasi dequeue dan enqueue dapat memerlukan waktu yang lebih lama jika tidak diimplementasikan dengan baik.

Percobaan :

1. Percobaan 1

```
2. class Queue {
```

```
3.   List<int> _elements = [];
4.   int _front = 0;
5.   int _rear = 0;
6.   int _maxQueue = 0;
7.
8.   void QueueOperation(int max) {
9.     _front = 0;
10.    _rear = -1;
11.    _elements = List<int>.filled(max, 0);
12.    _maxQueue = max - 1;
13.  }
14.
15.  bool isEmpty() {
16.    return _rear == -1 && _front == 0;
17.  }
18.
19.  bool isFull() {
20.    return _rear == _maxQueue - 1;
21.  }
22.
23.  void enqueue(int data) {
24.  if (isFull()) {
25.    print("Queue Overflow, tidak dapat mengisi data lagi");
26.  } else {
27.    _rear += 1;
28.    _elements[_rear] = data;
29.  }
30.}
31.
32.int dequeue() {
33.  int data = 0;
34.  if (isEmpty()) {
35.    print("Queue Underflow");
36.  } else {
37.    data = _elements[_front];
38.    _elements[_front] = 0;
39.    _front += 1;
40.  }
41.  return data;
42.}
43.
44.  void printQueue() {
45.    if (!isEmpty()) {
46.      print("Menampilkan urutan dari posisi tertinggi");
47.      for (int i = _rear; i > -1; i--) {
48.        print("Elemen ke-$i = ${_elements[i]}");
49.      }
50.    } else {
```

```
51.       print("Queue masih kosong");
52.     }
53.   }
54.}
55.
56.void main() {
57.   Queue q = Queue();
58.   q.QueueOperation(100);
59.   q.enqueue(10);
60.   q.enqueue(20);
61.   q.enqueue(30);
62.
63.   q.dequeue();
64.   q.printQueue();
65.}
```

Hasil run :

```
Menampilkan urutan dari posisi tertinggi
Elemen ke-2 = 30
Elemen ke-1 = 20
Elemen ke-0 = 0

Exited.
```

2. Percobaan ini hampir sama dengan sebelumnya, namun dengan style yang berbeda.

```
3. class Queue {
4.   int front = 0, rear = 0, size = 0;
5.   int capacity = 0;
6.   List<int> array = [];
7.
8.   void QueueOperation(int capacity) {
9.   this.capacity = capacity;
10.  front = size = 0;
11.  rear = capacity - 1;
12.  array = List<int>.filled(capacity, 0);
13.  }
14.
15.  bool isFull() {
16.    return (size == capacity);
17.  }
18.
19.  bool isEmpty() {
20.    return (size == 0);
21.  }
22.
23.  void enqueue(int item) {
```

```
    if (isFull()) return;
    rear = (rear + 1) % capacity;
    array[rear] = item;
    size++;
    print('$item enqueued to queue');
  }

  int dequeue() {
    if (isEmpty()) return -1;
    int item = array[front];
    front = (front + 1) % capacity;
    size--;
    return item;
  }

  int frontElement() {
    if (isEmpty()) return -1;
    return array[front];
  }

  int rearElement() {
    if (isEmpty()) return -1;
      return array[rear];
    }
  }
void main() {
  Queue q = Queue();
  q.QueueOperation(1000);
  q.enqueue(10);
  q.enqueue(20);
  q.enqueue(30);
  q.enqueue(40);

  print('${q.dequeue()} dequeued from queue\n');
  print('Front item is ${q.frontElement()}');
  print('Rear item is ${q.rearElement()}');
}
```

Hasil run :

```
10 enqueued to queue
20 enqueued to queue
30 enqueued to queue
40 enqueued to queue
10 dequeued from queue

Front item is 20
Rear item is 40

Exited.
```

Latihan :

1. Lakukan pembuatan program yang menerapkan Queue dengan menggunakan konsep Circular Array (Referensi ada di PPT, tinggal melanjutkan syntax programnya)!

```
2.  class CircularQueue<T> {
3.    List<T?> _array;
4.    int _front;
5.    int _rear;
6.    int _maxSize;
7.
8.    CircularQueue(this._maxSize)
9.        : _array = List<T?>.filled(_maxSize, null),
10.         _front = -1,
11.         _rear = -1;
12.
13.   bool isEmpty() {
14.     return _front == -1;
15.   }
16.
17.   bool isFull() {
18.     return (_rear + 1) % _maxSize == _front;
19.   }
20.
21.   void enqueue(T element) {
22.     if (isFull()) {
23.       print('Queue is full. Cannot enqueue $element.');
24.       return;
25.     }
26.     if (isEmpty()) {
27.       _front = 0;
28.     }
29.     _rear = (_rear + 1) % _maxSize;
30.     _array[_rear] = element;
31.     print('Enqueued: $element');
32.   }
33.
34.   T? dequeue() {
```

```
    if (isEmpty()) {
      print('Queue is empty. Cannot dequeue.');
      return null;
    }
    T? element = _array[_front];
    _array[_front] = null;
    if (_front == _rear) {
      _front = -1;
      _rear = -1;
    } else {
      _front = (_front + 1) % _maxSize;
    }
    print('Dequeued: $element');
    return element;
  }

  void display() {
    if (isEmpty()) {
      print('Queue is empty.');
      return;
    }
    print('Queue elements:');
    int i = _front;
    while (true) {
      print(_array[i]);
      if (i == _rear) break;
      i = (i + 1) % _maxSize;
    }
  }
}

void main() {
  CircularQueue<int> queue = CircularQueue<int>(5);

  queue.enqueue(10);
  queue.enqueue(20);
  queue.enqueue(30);
  queue.display();

  queue.dequeue();
  queue.display();

  queue.enqueue(40);
  queue.enqueue(50);
  queue.enqueue(60);
  queue.display();
}
```

Penjelasan :

1. **CircularQueue Class**:
   - Menggunakan array untuk menyimpan elemen queue.
   - Memiliki metode untuk memeriksa apakah queue kosong atau penuh, menambahkan elemen (enqueue), menghapus elemen (dequeue), dan menampilkan elemen dalam queue.
2. **enqueue**:
   - Menambahkan elemen ke queue. Jika queue penuh, akan menampilkan pesan.
3. **dequeue**:
   - Menghapus elemen dari queue. Jika queue kosong, akan menampilkan pesan.
4. **display**:
   - Menampilkan semua elemen dalam queue.
5. **Main Function**:
   - Menciptakan instansi dari CircularQueue dan melakukan beberapa operasi untuk menguji fungsionalitas.

Hasil run :

```
PS D:\dartsdk-windows-x64-release\sd\dart_application_1\bin> dart run latihan1_modul6.dart
Enqueued: 10
Enqueued: 20
Enqueued: 30
Queue elements:
10
20
30
Dequeued: 10
Queue elements:
20
30
Enqueued: 40
Enqueued: 50
Enqueued: 60
Queue elements:
20
30
40
50
60
PS D:\dartsdk-windows-x64-release\sd\dart_application_1\bin>
```

2.Buatlah sebuah program untuk antrian sebagaimana di dunia nyata, seperti antrian pada reservasi, antrian loket di bank, dan sebagainya. Silakan mengambil objek sesuai dengan preferensi masing-masing, namun dengan ketentuan:

- Menerapkan menu pilihan data antrian ditambah dan dikurangi dengan menggunakan perintah input oleh user (menggunakan Scanner/sejenisnya)
- Penambahan objek pada daftar antrian. Lakukan beberapa kali hingga antrian penuh. Ketika objek ditambahkan pada antrian yang penuh, maka program akan menampilkan keterangan antrian penuh
- Menampilkan isi antrian
- Satu persatu objek keluar hingga antrian kosong

```
import 'dart:io';

class Queue<T> {
  List<T?> _array;
  int _front;
  int _rear;
  int _maxSize;

  Queue(this._maxSize)
      : _array = List<T?>.filled(_maxSize, null),
        _front = -1,
        _rear = -1;

  bool isEmpty() => _front == -1;

  bool isFull() => (_rear + 1) % _maxSize == _front;

  void enqueue(T element) {
    if (isFull()) {
      print('Antrian penuh. Tidak bisa menambahkan $element.');
      return;
    }
    if (isEmpty()) {
      _front = 0;
    }
    _rear = (_rear + 1) % _maxSize;
    _array[_rear] = element;
    print('Ditambahkan ke antrian: $element');
  }

  T? dequeue() {
    if (isEmpty()) {
      print('Antrian kosong. Tidak bisa mengeluarkan elemen.');
      return null;
    }
    T? element = _array[_front];
```

```
    _array[_front] = null;
    if (_front == _rear) {
      _front = -1;
      _rear = -1;
    } else {
      _front = (_front + 1) % _maxSize;
    }
    print('Dikeluarkan dari antrian: $element');
    return element;
  }

  void display() {
    if (isEmpty()) {
      print('Antrian kosong.');
      return;
    }
    print('Isi antrian:');
    int i = _front;
    while (true) {
      print(_array[i]);
      if (i == _rear) break;
      i = (i + 1) % _maxSize;
    }
  }
}

void main() {
  Queue<String> queue = Queue<String>(5); // Ukuran antrian 5
  while (true) {
    print('\nMenu:');
    print('1. Tambah ke antrian');
    print('2. Keluarkan dari antrian');
    print('3. Tampilkan isi antrian');
    print('4. Keluar');
    stdout.write('Pilih opsi: ');

    String? choice = stdin.readLineSync();

    switch (choice) {
      case '1':
        stdout.write('Masukkan nama untuk ditambahkan ke antrian: ');
        String? name = stdin.readLineSync();
        if (name != null) {
          queue.enqueue(name);
        }
        break;
      case '2':
        queue.dequeue();
```

```
        break;
      case '3':
        queue.display();
        break;
      case '4':
        print('Keluar dari program.');
        return;
      default:
        print('Pilihan tidak valid. Silakan coba lagi.');
    }
  }
}
```

Penjelasan:

1.Queue Class:

- Mengelola elemen dalam antrian menggunakan array.
- Metode untuk menambah (enqueue), mengurangi (dequeue), dan menampilkan (display) elemen dalam antrian.

2.**Menu Utama**:

- Menggunakan loop untuk memberikan pilihan kepada pengguna.
- Pengguna dapat menambahkan nama, mengeluarkan nama dari antrian, menampilkan isi antrian, atau keluar dari program.

3.**Input Pengguna**:

- Menggunakan stdin.readLineSync() untuk mengambil input dari pengguna.

Hasil run :

```
4. Keluar
Pilih opsi: 1
Masukkan nama untuk ditambahkan ke antrian: dzikra
Ditambahkan ke antrian: dzikra

Menu:
1. Tambah ke antrian
2. Keluarkan dari antrian
3. Tampilkan isi antrian
4. Keluar
Pilih opsi: 3
Isi antrian:
diana
dzikra

Menu:
1. Tambah ke antrian
2. Keluarkan dari antrian
3. Tampilkan isi antrian
4. Keluar
Pilih opsi: 4
Keluar dari program.
PS D:\dartsdk-windows-x64-release\sd\dart_application_1\bin>
```

Kesimpulan :

1. **Stack** beroperasi dengan prinsip LIFO (Last In First Out), sedangkan **Queue** menggunakan prinsip FIFO (First In First Out). Kedua prinsip ini menentukan bagaimana elemen dimasukkan dan dikeluarkan dari struktur data.
2. **Queue:** Digunakan dalam manajemen antrian pelanggan, penjadwalan tugas, dan pengelolaan proses dalam sistem operasi**.**

Referensi :

https://api.flutter.dev/flutter/widgets/Stack-class.html

https://api.flutter.dev/flutter/dart-collection/Queue-class.html

https://www.geeksforgeeks.org/queues-in-dart/