BAB VIII

STACK DAN QUEUE

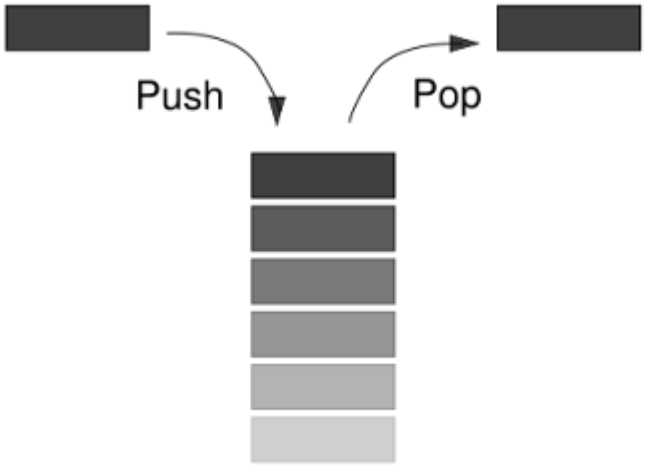
8.1 Tujuan

1. Praktikan mampu memahami dasar penggunaan stack dan queue pada Java,C++, PHP dan Python.
2. Praktikan mampu membedakan konsep dasar stack dan queue dari bahasa pemrograman yang berbeda (Java, C++, PHP, dan Python).
3. Praktikan mampu mengimplementasikan stack dan queue dalam pemrograman Java, C++, PHP, dan Python.

8.2 Dasar Teori

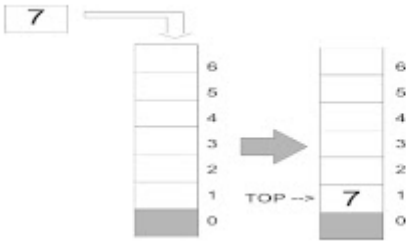
8.2.1 Stack

Stack atau tumpukan dapat diartikan sebagai suatu kumpulan data yang seolah-olah terlihat seperti ada data yang diletakkan di atas data yang lain. Kaidah utama dalam konsep stack adalah LIFO yang merupakan singkatan dari *Last In First Out*, artinya adalah data yang terakhir kali dimasukkan atau disimpan, maka data tersebut adalah yang pertama kali akan diakses atau dikeluarkan. Berikut ilustrasi kerja sebuah stack:

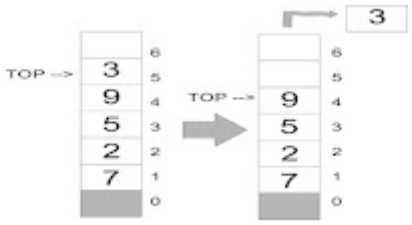


Sebuah struktur data dari sebuah *stack* setidaknya harus mengandung dua buah variabel, misalnya variabel *top* yang akan berguna sebagai penanda bagian atas tumpukan dan *array* data dari yang akan menyimpan data-data yang dimasukkan ke dalam *stack* tersebut. Deklarasi struktur data dari sebuah *stack*:
Operasi-operasi dasar dalam *stack* ada 2 yaitu operasi *push* dan *pop*:

1. Operasi *push*, berfungsi untuk memasukkan sebuah nilai atau data ke dalam *stack*. Sebelum sebuah nilai atau data dimasukkan ke dalam *stack*, prosedur ini terlebih dahulu akan menaikkan posisi *top* satu *level* ke atas. Berikut ilustrasi kerja pada operasi *push*:



2. Operasi *pop*, berfungsi untuk mengeluarkan atau menghapus nilai terakhir (yang berada pada posisi paling atas) dari *stack*, dengan cara menurunkan nilai *top* satu *level* ke bawah. Berikut ilustrasi kerja pada operasi *pop*:



- Algoritma *Stack*

a. Operasi *Push*

Operasi *push* digunakan untuk menambahkan sebuah elemen baru ke atas tumpukan. Operasi ini dapat dilakukan jika tumpukan tidak penuh. Algoritma operasi *push* pada *stack* adalah sebagai berikut:

- Menentukan kondisi tumpukan, apakah tumpukan dalam keadaan kosong atau tidak.
- Jika kosong maka mendeklarasikan data baru yang akan dimasukkan ke dalam tumpukan.
- Memasukkan nilai data yang baru.
- Melakukan perulangan untuk memasukkan data hingga batas penuh tumpukan.
- Jika tumpukan sudah penuh maka selesai.

b. Operasi *Pop*

Operasi *pop* digunakan untuk menghapus elemen teratas dari tumpukan. Algoritma operasi *pop* pada *stack* adalah sebagai berikut:

- Melakukan pengecekan kondisi antrian, ada isi data atau tidak.

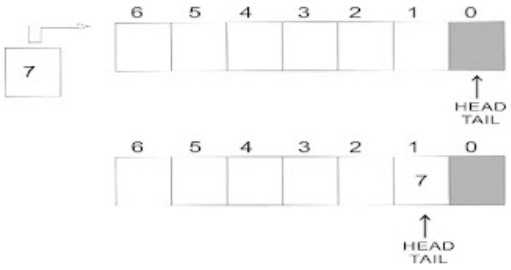
- Jika terdapat data pada antrian, maka lakukan penghapusan data dengan cara memindahkan *head* (elemen teratas tumpukan) ke elemen dibawahnya.
- Kemudian menghapus elemen teratas tumpukan.
- Jika tidak terdapat data pada tumpukan maka selesai.

8.2.2 Queue

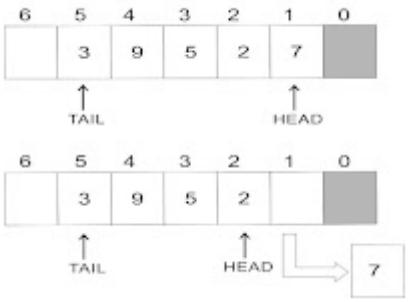
Queue atau antrian merupakan struktur data linear dimana penambahan komponen dilakukan disatu ujung, sementara pengurangan dilakukan diujung lain. Kaidah utama dalam konsep *queue* adalah FIFO yang merupakan singkatan dari *First In First Out*, artinya adalah data yang pertama kali dimasukkan atau disimpan, maka data tersebut adalah yang pertama kali akan diakses atau dikeluarkan.

Sebuah *queue* di dalam program komputer dideklarasikan sebagai sebuah tipe bentukan baru. Sebuah struktur data dari sebuah *queue* setidaknya harus mengandung dua tiga variabel, yakni variabel *head* yang akan berguna sebagai penanda bagian depan antrian, variabel *tail* yang akan berguna sebagai penanda bagian belakang antrian dan *array* dari yang akan menyimpan data-data yang dimasukkan ke dalam *queue* tersebut. Deklarasi struktur data dari sebuah *queue*:
Operasi-operasi dasar dalam *stack* ada 2 yaitu operasi *enqueue* dan *dequeue*:

1. Operasi *enqueue*, digunakan untuk memasukkan sebuah data atau nilai ke dalam *queue*. Pada proses *enqueue*, *tail* -lah yang berjalan seiring masuknya data baru ke dalam antrian, sedangkan *head* akan tetap pada posisi ke-1.



2. Operasi dequeue, digunakan untuk menghapuskan sebuah data atau nilai yang paling awal masuk ke dalam queue. Operasi ini menaikkan nilai head satu level



- Algoritma Queue

a. Operasi *Enqueue*

Operasi *push* pada antrian disebut juga *enqueue*. Operasi ini digunakan untuk menambah sebuah elemen baru. Elemen baru akan dimasukkan ke belakang antrian. Algoritma operasi *push* pada *queue* adalah sebagai berikut:

- Menentukan kondisi antrian, apakah antrian dalam keadaan kosong atau tidak.
- Jika kosong maka mendeklarasikan data baru yang akan dimasukkan ke dalam antrian.
- Memasukkan nilai data yang baru.
- Melakukan perulangan untuk memasukkan data hingga batas penuh antrian dengan cara menempatkan penunjuk *front* menunjuk ke elemen terdepan (*head*) pada antrian dan *rear* menunjuk ke elemen baru yang ditambahkan dan nilai *count* bertambah satu.
- Jika antrian sudah penuh maka selesai.

b. Operasi *Dequeue*

Operasi *pop* pada antrian disebut juga *dequeue*. Operasi ini digunakan untuk menghapus elemen pada antrian. Elemen yang akan dihapus adalah elemen yang terletak paling depan pada antrian. Algoritma operasi *pop* pada *queue* adalah sebagai berikut:

- Melakukan pengecekan kondisi antrian, ada isi data atau tidak.
- Jika terdapat data pada antrian, maka lakukan penghapusan data dengan cara memindahkan *head* (elemen terdepan antrian) ke elemen setelahnya atau membuat penunjuk *front* menunjuk ke elemen setelah elemen terdepan pada *queue*.
- Kemudian menghapus elemen terdepan antrian dan nilai *count* antrian berkurang satu.
- Jika tidak terdapat data pada antrian maka selesai.

8.3 Percobaan

8.3.1 Bahasa Pemrograman Java

Stack pada Java

Main :

```
package com.strukturdata.stack;

public class MainStack {

    public static void main(String[] args) {

        Stack stack = new Stack(5);

        stack.push("Apel");
        stack.push("Belimbing");
        stack.push("Manggis");
        stack.push("Durian");
        stack.push("Sirsak");
        stack.peek();
        stack.count();
        stack.printStack();

        System.out.println();

        stack.pop();
        stack.pop();
        stack.peek();
        stack.count();
        stack.printStack();

        stack.clear();
        stack.printStack();

    }

}
```

Class Stack :

```
package com.strukturdata.stack;

public class Stack {

    private int capacity;
    private Object[]data;
    private int top;
    private int count;


    public Stack(int size){
        top = -1;
        data = new Object[size];
```

```
        capacity = size;
        count = 0;
    }

    public boolean isFull(){
        return top == capacity - 1;
    }

    public boolean isEmpty(){
        return top == -1;
    }

    public void push(Object x){
        if(isFull()){
            System.out.println("Stack Penuh!!!");
        }else{
            System.out.println("Memasukkan " + x + " Ke Dalam
Stack");
            data[++top] = x;
        }
    }

    public Object pop(){
        if(isEmpty()){
            System.out.println("Stack Kosong!!!");
            return 0;
        }
        System.out.println("Mengeluarkan " + data[top] + " Dari
Stack");
        return data[top--];
    }

    public void printStack(){
        System.out.print("Data Dalam Stack : ");
        if(isEmpty()){
            System.out.println("Stack Kosong!!!");
        }else{
            for(int i = top; i >= 0; i--){
                System.out.print(data[i] + " ");
            }
            System.out.println();
        }
    }




    public void peek(){
        if(isEmpty()){
            System.out.println("Stack Kosong!!!");
        }else{
            System.out.println("Data Teratas : " + data[top]);
        }
    }

    public void clear(){
```

```
        if(isEmpty()){
            System.out.println("Stack Kosong!!!");
        }else{
            top = -1;
        }
    }

    public void count(){
        System.out.println("Jumah Data Dalam Stack : " + (top +
1));
    }

}
```

1. Push, digunakan untuk memasukkan data ke dalam Stack
2. Pop, digunakan untuk mengeluarkan data teratas dari Stack
3. Peek, digunakan untuk melihat data yang berada di posisi paling atas
4. Count, digunakan untuk mengetahui jumlah isi data pada Stack
5. Clear, digunakan untuk mengapus seluruh data yang ada pada Stack
6. isEmpty, digunakan untuk mengecek apakah Stack kosong atau tidak
7. isFull, digunakan untuk mengecek apakah Stack penuh atau tidak
8. printStack, digunakan untuk menampilkan semua data pada Stack
9. Stack, karena bersifat menyimpan data maka memerlukan struktur data yang lain, dalam kasus menggunakan **Array**.

Queue pada Java

Main :

```
package com.strukturdata.queue;

public class MainQueue {

    public static void main(String[] args) {

        Queue queue = new Queue(4);

        queue.enqueue(1);
        queue.enqueue(2);
        queue.enqueue(3);
        queue.enqueue(4);
        queue.peek();
        queue.count();
        queue.printQueue();

        System.out.println();
```

```
        queue.dequeue();
        queue.dequeue();
        queue.peek();
        queue.count();
        queue.printQueue();

    }
}
```

Class Queue :

```
package com.strukturdata.queue;

public class Queue {

    private Object[] data;
    private int front;
    private int rear;
    private int capacity;
    private int count;


    public Queue(int size){
        data = new Object[size];
        capacity = size;
        rear = -1;
        front = 0;
        count = 0;
    }

    public boolean isFull(){
        return count == capacity;
    }

    public boolean isEmpty(){
        return count == 0;
    }

    public void enqueue(Object x){
        if(isFull()){
            System.out.println("Queue Penuh!!!");
        }else{
            System.out.println("Memasukkan " + x + " Ke Dalam
Queue");

            rear = (rear + 1) % capacity;
            data[rear] = x;
            count++;
        }
    }

    public Object dequeue(){
        if(isEmpty()){
            System.out.println("Queue Kosong!!!");
```

```
            return 0;
        }

        Object x = data[front];

        System.out.println("Mengeluarkan " + x + " Dari Queue");

        front = (front + 1) % capacity;
        count--;

        return x;
    }

    public void printQueue(){
        if(isEmpty()){
            System.out.println("Queue Kosong!!!");
        }else{
            System.out.print("Data Dalam Queue : ");
            for(int i = 0; i < count ; i++){
                System.out.print(data[(front + i) % capacity] +
" ");
            }

            System.out.println();
        }
    }

    public void peek(){
        if(isEmpty()){
            System.out.println("Queue Kosong!!!");
        }else{
            System.out.println("Elemen Depan : " + data[front]);
        }
    }

    public void count(){
        System.out.println("Jumlah Data Dalam Queue : " +
count);
    }

}
```

1. Enqueue, digunakan untuk memasukkan data ke dalam Queue
2. Dequeue, digunakan untuk mengeluarkan data terdepan dari Queue
3. Peek, digunakan untuk melihat data yang berada di posisi paling depan
4. Count, digunakan untuk mengetahui jumlah isi data pada Queue
5. isEmpty, digunakan untuk mengecek apakah Queue kosong atau tidak
6. isFull, digunakan untuk mengecek apakah Queue penuh atau tidak
7. printQueue, digunakan untuk menampilkan seluruh data pada Queue

8.3.2 Bahasa Pemrograman C++

Stack pada C++

Stack dapat diilustrasikan sebagai tumpukan data dimana data pertama yang masuk maka data itu berada di paling bawah, untuk memasukkan data dapat digunakan perintah push, misal kita melakukan push 1, push 2, push 3, push 4, maka yang akan dikeluarkan nanti yaitu 1,2,3,4 berurutan dari bawah. Dari data push tadi, jika ingin mengeluarkan data 1, maka proses yang dilakukan pop 4, pop 3, pop 2, pop 1. Untuk lebih jelas berikut contoh list code *stack* dan *queue* pada c++.

```
//Preprosesor
#include <iostream>
#include <conio.h>
#include <stdlib.h>
using namespace std;

//Deklarasi stack dengan struct dan array
struct STACK
{
    int data [5];
    int top;
};

//deklarasi variabel tumpukan dari struct
STACK tumpukan;

//deklarasi fungsi operasi stack
void inisialisasi();
int IsEmpty();
int IsFull();
void push (int data);
void pop ();

//fungsi main program
main ()
{
    system("cls");
    int pilih, input, i;
    inisialisasi();
    do{
        cout<<"Masukkan Angka"<<endl;
        cout<<"1. Push data"<<endl;
        cout<<"2. Pop Data"<<endl;
        cout<<"3. Print Data"<<endl;
        cout<<"4. Clear Data"<<endl;
        cout<<endl;
        cout<<"Pilih : ";cin>>pilih;
        switch(pilih)
        {
            case 1:
            {
                if(IsFull()==1)
```

```
                {
                    cout<<"Tumpukan penuh !";
                }
                else
                {
                    cout<<"Data yang akan di push :
";cin>>input;
                    push(input);
                }
                cout<<endl;
                getch();
                break;
            }
            case 2:
            {
                if(IsEmpty()==1)
                {
                    cout<<"Tumpukan Kosong !";
                }
                else
                {
                    cout<<"Data yang akan di Pop =
"<<tumpukan.data[tumpukan.top]<<endl;
                    pop();
                }
                cout<<endl;
                getch();
                break;
            }
            case 3:
            {
                if(IsEmpty()==1)
                {
                    cout<<"Tumpukan Kosong !"<<endl;
                }
                else
                {
                    cout<<"Data : "<<endl;
                    for(i=0; i<=tumpukan.top; i++)
                    {
                        cout<<tumpukan.data[i]<<" ";
                    }
                }
                cout<<endl;
                getch();
                break;
            }
            case 4:
            {
                inisialisasi();
                cout<<"Tumpukan Kosong !"<<endl;
                cout<<endl;
                getch();
                break;
            }
            default:
```

```
            {
                cout<<"Tidak ada dalam pilih"<<endl;
            }
        }
    } while (pilih!=0);
}

//fungsi inisialisasi stack = kosong
void inisialisasi()
{
    tumpukan.top=-1;
}

//fungsi mengecheck apakah stack kosong
int IsEmpty()
{
    if(tumpukan.top==-1)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

//fungsi mengecheck apakah stack penuh
int IsFull()
{
    if (tumpukan.top==5-1)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

//fungsi untuk menyisipkan data ke stack
void push(int data)
{
    tumpukan.top++;
    tumpukan.data[tumpukan.top]=data;
}

//fungsi untuk mengeluarkan data dari stack
void pop()
{
    tumpukan.top=tumpukan.top-1;
    if(tumpukan.top<0)
    {
        tumpukan.top=-1;
    }
}
```

Queue pada C++

Berbeda dengan *stack*, konsep dasar *queue* yaitu antrian jadi yang pertama dimasukan berada di paling atas. Bila dalam *queue* apabila kita menuliskan enqueue 1, enqueue 2, enqueue 3, enqueue 4. Dari enqueue tadi maka hasilnya yaitu urut 1,2,3,4 dari atas. Sedangkan fungsi dequeue pada *queue* sama seperti fungsi pop pada *stack*. Apabila melakukan dequeue, maka hasilnya 2,3,4.

```
#include <iostream>
#define MAX 20 //maksimum data queue
using namespace std;

//Deklarasi struct antrian
struct Queue {
      int front, rear, data[MAX];
}Q;

//cek apakah antrian penuh
bool isFull() {
      return Q.rear == MAX;
}

//cek apakah antrian kosong
bool isEmpty() {
      return Q.rear == 0;
}

//Menampilkan Queue
void printQueue() {
      if (isEmpty()) {
        cout << "Antrian kosong"<<endl;
      }
      else {
            cout << "QUEUE : ";
            for (int i = Q.front; i < Q.rear; i++)
            //menambahkan koma jika data tidak terdapat di
antrian pertama
                  cout << Q.data[i] << ((Q.rear-1 == i) ? "" :
",");
            cout << endl;
  }
}
//manambahkan data ke antrian
void enqueue() {
      if (isFull())
      {
            cout << "Antrian penuh!"<<endl;
      }
      else {
            int data;
            //menambahkan data ke antrian
            cout << "Masukkan Data : ";cin >> data;
            Q.data[Q.rear] = data;
```

```
            //menempatkan tail pada elemen data terakhir yang
ditambahkan
            Q.rear++;
            cout << "Data ditambahkan\n";
            printQueue();
      }
}

// mengambil data dari antrian
void dequeue() {
      if (isEmpty())
      {
            cout << "Antrian masih kosong"<<endl;
      }
      else{
            cout << "Mengambil data \"" << Q.data[Q.front] <<
"\"..." << endl;
            //menggeser antrian data ke head
            for (int i = Q.front; i < Q.rear; i++)
                  Q.data[i] = Q.data[i + 1];
            //menempatkan tail pada data terakhir yang digeser
            Q.rear--;
            printQueue();
      }
}

int main() {
      int choose;
      do
      {
            //Tampilan menu
            cout << "--------------------\n"
                  << "   Menu Pilihan\n"
                  << "--------------------\n"
                  << " [1] Enqueue \n"
                  << " [2] Dequeue\n"
                  << " [3] Keluar \n\n"
                  << "--------------------\n"
                  << "Masukkan pilihan : "; cin >> choose;
            switch (choose)
            {
            case 1:
                  enqueue();
                  break;
            case 2:
                  dequeue();
                  break;
            default:
                  cout << "Pilihan tidak tersedia";
                  break;
            }
      } while (choose !=3);
      return 0;
}
```

8.3.3 Bahasa Pemograman Python

Stack pada Python

Source code sederhana dari *stack.*

```
fruits = []

#menambahkan value pada array
fruits.append('banana')
fruits.append('grapes')
fruits.append('mango')
fruits.append('orange')

print(fruits)

#Melakukan Pop pada array, dan mengambil value dari first item
first_item = fruits.pop(0)
print(first_item)

#Melakukan Pop pada array, dan mengambil value dari first item
first_item = fruits.pop(0)
print(first_item)

print(fruits)
```

Untuk contoh lain dari *stack* pada Python:

```
count = 0
size = -1
stack = []
while count == 0:
  print("---Ini Stack---")
  print("1. PUSH Item")
  print("2. POP Item")
  print("3. Lihat Daftar Data")
  print("4. Lihat Data Teratas")
  print("5. Keluar")
  pilihan = int(input("Masukkan Pilihan : "))
  if pilihan == 1:
    data = int(input("Data yang ditambahkan : "))
    stack.append(data)
    size+=1
    print ("")
  if pilihan == 2:
    stack.pop()
    size-=1
    print ("")
  if pilihan == 3:
    print(stack)
    print ("")
  if pilihan == 4:
    print(stack[size])
    print ("")
  if pilihan == 5:
    count = 1
```

Program *stack* diatas terdapat 5 menu utama yaitu, *push*, *pop*, lihat daftar data, lihat

data teratas, dan keluar. Push berfungsi untuk menambahkan data ke dalam tumpukan, pop untuk mengurangi data dari data teratas, lihat daftar untuk melihat daftar data, lihat data teratas untuk melihat daftar data yang ada di urutan atas, dan keluar untuk menghentikan jalannya program. Di dalam program terdapat *while* yang berfungsi untuk selalu menampilkan menu utama setelah pengguna memberikan instruksi pada program.

Queue pada Python

Source code sederhana dari *queue*.

```
from collections import deque

numbers = deque()

numbers.append(99)
numbers.append(15)
numbers.append(82)
numbers.append(50)
numbers.append(47)

print(numbers)

last_item = numbers.pop()
print(last_item)
print(numbers)

first_item = numbers.popleft()
print(first_item)
print(numbers)
```

Untuk contoh lain dari *queue* pada Python:

```
from collections import deque
count = 0
queue = deque()
while count == 0:
    print("---Ini Queue---")
    print("1. Enqueue Item")
    print("2. Dequeue Item")
    print("3. Lihat Daftar Data")
    print("4. Lihat Data Teratas")
    print("5. Keluar")
    pilihan = int(input("Masukkan Pilihan : "))
    if pilihan == 1:
        data = int(input("Data yang ditambahkan : "))
        queue.append(data)
        print ("")
    if pilihan == 2:
        queue.popleft()
```

```
        print ("")
    if pilihan == 3:
        print(queue)
        print ("")
    if pilihan == 4:
        print(queue[0])
        print ("")
    if pilihan == 5:
        count = 1
```

Diatas merupakan hasil dari program *queue* menggunakan bahasa Python. Program diatas terdapat 5 menu utama. *Enqueue* untuk menambahkan ke antrian. *Dequeue* untuk mengurangi data dari antrian. Karna prinsip dari Queue adalah *first in first out*, maka ketika melakukan *dequeue* maka data yang terhapus adalah data yang pertama dimasukkan. Lihat daftar untuk melihat daftar data, lihat data teratas untuk melihat daftar data yang ada di urutan atas, dan keluar untuk menghentikan jalannya program. Di dalam program terdapat *while* yang berfungsi untuk selalu menampilkan menu utama setelah pengguna memberikan instruksi pada program.

8.3.4 Bahasa Pemograman PHP

Stack pada PHP

Stack dapat diilustrasikan sebagai tumpukan data dimana data pertama yang masuk maka data itu berada di paling bawah, untuk memasukkan data dapat digunakan perintah push, misal kita melakukan *push* 1, *push* 2, *push* 3, *push* 4, maka yang akan dikeluarkan nanti yaitu 1,2,3,4 berurutan dari bawah. Dari data *push* tadi, jika ingin mengeluarkan data 1, maka proses yang dilakukan *pop* 4, *pop* 3, *pop* 2, *pop* 1. Untuk lebih jelas berikut contoh list *code stack* dan *queue* pada PHP.

```
<h2>PHP Stack</h2>
<form action="" method="POST">
    ==========[ PUSH ]==========<br><br>
    <label for="data">Data 1: </label>
    <input type="text" name="data1" /><br>
    <label for="data">Data 2: </label>
    <input type="text" name="data2" /><br>
    <label for="data">Data 3:</label>
    <input type="text" name="data3" />
    <br><br>
    ===========[ POP ]==========<br><br>
    <label for="data">Amount: </label>
    <input type="number" name="jmlpop" />
    <br><br>
```

```
    <input name="submit" type="submit" value="SUBMIT" />
    <br><br>
    ==========================
</form>

<?php
// start a session
$stack = new SplQueue();
$stack->push(1);
$stack->push(2);
$stack->push(3);
$stack->push(4);
$stack->push(5);
$stack->push(6);
$stack->push(7);
$stack->push(8);

print("Data lama[" . sizeof($stack) . "]: ");
for ($i = 0; $i < sizeof($stack); $i++) {
    print($stack[$i] . " ");
}

if (isset($_POST['submit'])) {

    /////// PUSH ///////
    if ($_POST['data1'] != NULL || $_POST['data2'] != NULL ||
$_POST['data3'] != NULL) {
        $data[0] = $_POST['data1'];
        $data[1] = $_POST['data2'];
        $data[2] = $_POST['data3'];
        for ($i = 0; $i < 3; $i++) {
            if ($data[$i] != null) {
                $stack->push($data[$i]);
            }
        }

        print("<br><br>==========[ PUSH ]==========<br><br>");

        print("Data baru[" . sizeof($stack) . "]: ");
        for ($i = 0; $i < sizeof($stack); $i++) {
            print($stack[$i] . " ");
        }
    }

    /////// POP ///////
    if ($_POST['jmlpop'] != NULL) {
        print("<br><br>==========[ POP ]==========<br><br>");
        $jmlpop = $_POST['jmlpop'];
        for ($i = 0; $i < $jmlpop; $i++) {
            $stack->pop();
        }
        print("Data baru[" . sizeof($stack) . "]: ");
        for ($i = 0; $i < sizeof($stack); $i++) {
            print($stack[$i] . " ");
        }
    }
```

```
    /////// RESULT ///////
    print("<br><br>==========================<br><br>");
    print("Data teratas: " . $stack->top());
}
```

Source code di atas merupakan pengimplementasian Stack pada PHP. Untuk membuat Stack maupun Queue pada PHP diperlukan *library* SplQueue. Penggunaan *method* push dan pop pada *library* SplQueue tetap bekerja sebagaimana Stack pada umumnya. *Method* push digunakan untuk menambahkan data sedangkan *method* pop digunakan untuk menghapus data teratas pada Stack. Pada *source code* di atas diperlukan HTML Form agar pengguna dapat melakukan *input*. Untuk mendapatkan data teratas dari Stack maka digunakan *method top*.

Queue pada PHP

```
<h2>PHP Queue</h2>
<form action="" method="POST">
    ==========[ ENQUEUE ]==========<br><br>
    <label for="data">Data 1: </label>
    <input type="text" name="data1" /><br>
    <label for="data">Data 2: </label>
    <input type="text" name="data2" /><br>
    <label for="data">Data 3:</label>
    <input type="text" name="data3" />
    <br><br>
    ===========[ DEQUEUE ]==========<br><br>
    <label for="data">Amount: </label>
    <input type="number" name="jmldeq" />
    <br><br>
    <input name="submit" type="submit" value="Submit" />
    <br><br>
    ==========================
</form>

<?php

$queue = new SplQueue();
$queue->enqueue('Messi');
$queue->enqueue('Ronaldo');
$queue->enqueue('Benzema');
$queue->enqueue('Haaland');
$queue->enqueue('Mbappe');

print("Data lama[" . sizeof($queue) . "]: ");
for ($i = 0; $i < sizeof($queue); $i++) {
    print($queue[$i] . " ");
}
```

```
if (isset($_POST['submit'])) {

    /////// ENQUEUE ///////
    if ($_POST['data1'] != NULL || $_POST['data2'] != NULL ||
$_POST['data3'] != NULL) {
        $data[0] = $_POST['data1'];
        $data[1] = $_POST['data2'];
        $data[2] = $_POST['data3'];
        for ($i = 0; $i < 3; $i++) {
            if ($data[$i] != null) {
                $queue->enqueue($data[$i]);
            }
        }

        print("<br><br>==========[                          ENQUEUE
]==========<br><br>");

        print("Data baru[" . sizeof($queue) . "]: ");
        for ($i = 0; $i < sizeof($queue); $i++) {
            print($queue[$i] . " ");
        }
    }

    /////// DEQUEUE ///////
    if ($_POST['jmldeq'] != NULL) {
        print("<br><br>==========[                          DEQUEUE
]==========<br><br>");
        $jmldeq = $_POST['jmldeq'];
        for ($i = 0; $i < $jmldeq; $i++) {
            $queue->dequeue();
        }
        print("Data baru[" . sizeof($queue) . "]: ");
        for ($i = 0; $i < sizeof($queue); $i++) {
            print($queue[$i] . " ");
        }
    }

    /////// RESULT ///////
    print("<br><br>==========================<br><br>");
    print("Data terdepan : " . $queue->bottom());
}
```

Source code di atas merupakan pengimplementasian Queue pada PHP. Untuk membuat Stack maupun Queue pada PHP diperlukan *library* SplQueue. Untuk menambahkan data pada Queue digunakan *method* enqueue. Sedangkan untuk menghapus data pertama pada Queue digunakan *method* dequeue. Pada *source code* di atas diperlukan HTML Form agar pengguna dapat melakukan *input*. Untuk mendapatkan data pertama atau terdepan dari Queue maka digunakan *method bottom*.