

# LAPORAN PRAKTIKUM ▼

## ALGORITMA DAN STRUKTUR DATA



IKA SULASTRI NINGSIH▼

240306030 ▼

[github.com/IKASULASTRININGSIH](https://github.com/IKASULASTRININGSIH)▼

Program Studi Teknologi Informasi  
Fakultas Dakwah dan Ilmu Komunikasi  
Universitas Islam Negeri Mataram  
2024

Pertemuan	9
Topik	Stack dan queue
Repository	
Tanggal	26 november 2024

#### A. Tujuan

Tujuan praktikum stack dan queue

1. Tujuan pratikum stack dan queue adalah untuk memahami dan mengimplementasikan dua struktur data dasar yang penting dalam pemrograman komputer, yaitu stack (tumpukan) dan queue (antrian). Tujuan lebih spesifik meliputi:
  2. 1. \*Memahami Konsep Dasar Stack dan Queue:\*
    - \*Stack:\* Struktur data yang mengikuti prinsip LIFO (Last In, First Out), di mana elemen terakhir yang dimasukkan akan menjadi elemen pertama yang dikeluarkan.
    - 3. - \*Queue:\* Struktur data yang mengikuti prinsip FIFO (First In, First Out), di mana elemen pertama yang dimasukkan akan menjadi elemen pertama yang dikeluarkan.
    - 4. 2. \*Mengimplementasikan Stack dan Queue:\*
    - 5. - Mempelajari cara membuat stack dan queue menggunakan array atau linked list.
    - 6. - Memahami operasi dasar pada kedua struktur data ini, seperti:
      7. - \*Push\* (untuk stack) dan \*enqueue\* (untuk queue) untuk menambahkan elemen.
      8. - \*Pop\* (untuk stack) dan \*dequeue\* (untuk queue) untuk menghapus elemen.
      9. - \*Peek\* untuk melihat elemen teratas (pada stack) atau elemen paling depan (pada queue) tanpa menghapusnya.
    - 3. \*Meningkatkan Kemampuan Pemecahan Masalah:\*

10. - Mempelajari bagaimana stack dan queue digunakan untuk memecahkan masalah tertentu dalam pengolahan data, misalnya, dalam algoritma pengurutan, pemrograman dinamis, dan masalah terkait graf.
11. - Memahami penerapan stack dalam rekursi dan proses pengolahan data berurutan.
12. 4. \*Menerapkan dalam Program Nyata:\*
13. - Menggunakan stack dan queue dalam aplikasi dunia nyata, seperti dalam penjadwalan tugas (queue), atau dalam algoritma seperti pencarian kedalaman pertama (DFS) yang menggunakan stack.
14. Dengan pratikum ini, diharapkan mahasiswa atau peserta dapat lebih memahami konsep dan aplikasi kedua struktur data tersebut dalam pengembangan perangkat lunak.
15. Merancang dan membuat flowchart sebagai representasi visual dari algoritma
16. Mengenal simbol-simbol standar dalam flowchart

## B. Requirement

1. Sistem Operasi yang digunakan : Windows 10 vers
2. Browser : Google Chrome Version  
129.0.6668.100 (Official Build)  
(x86\_64)
3. Tools yang digunakan : Visual Studio Code

## C. Dasar Teori

### Dasar Teori Stack dan Queue

#### Stack

Stack adalah struktur data linear yang mengikuti prinsip Last In, First Out (LIFO). Ini berarti elemen yang terakhir dimasukkan ke dalam stack akan menjadi elemen pertama yang dikeluarkan. Operasi dasar yang dapat dilakukan pada stack meliputi:

Push: Menambahkan elemen baru ke puncak stack.

Pop: Menghapus elemen dari puncak stack.

Peek/Top: Mengakses elemen di puncak stack tanpa menghapusnya.

IsEmpty: Memeriksa apakah stack kosong.

### Queue

Queue adalah struktur data linear yang mengikuti prinsip First In, First Out (FIFO).

Ini berarti elemen yang pertama kali dimasukkan ke dalam queue akan menjadi elemen pertama yang dikeluarkan. Operasi dasar yang dapat dilakukan pada queue meliputi:

Enqueue: Menambahkan elemen baru ke bagian belakang queue.

Dequeue: Menghapus elemen dari bagian depan queue.

Front/Peek: Mengakses elemen di bagian depan queue tanpa menghapusnya.

IsEmpty: Memeriksa apakah queue kosong.

### Implementasi

Stack: Dapat diimplementasikan menggunakan array atau linked list. Implementasi menggunakan array biasanya lebih efisien dalam hal penggunaan memori.

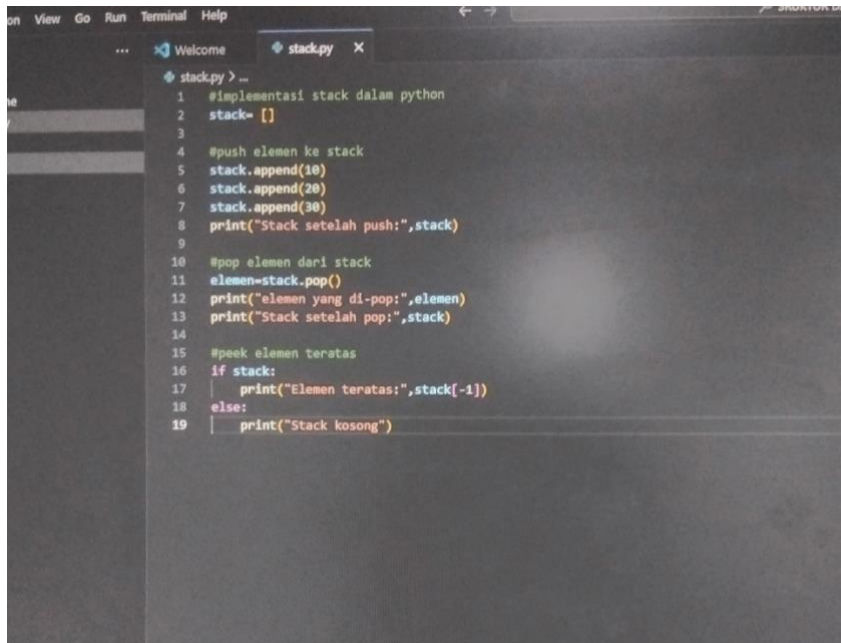
Queue: Biasanya diimplementasikan menggunakan array atau linked list, di mana penambahan dilakukan di satu ujung dan penghapusan dilakukan di ujung lainnya.

### Aplikasi

Stack: Digunakan dalam pemrograman rekursif, penelusuran depth-first dalam graf, dan manajemen tumpukan dalam sistem operasi.

Queue: Digunakan dalam penjadwalan proses dalam sistem operasi, simulasi antrian dalam permainan, dan manajemen print queue

## D. Implementasi

A screenshot of a code editor window with a dark theme. The editor has tabs at the top, with 'stack.py' selected. The code is written in Python and implements a stack using a list. It includes comments in Indonesian explaining each step: pushing elements (10, 20, 30), printing the stack after push, popping an element, printing the popped element and the stack after pop, and peeking at the top element. The code is as follows:

```
1 #implementasi stack dalam python
2 stack= []
3
4 #push elemen ke stack
5 stack.append(10)
6 stack.append(20)
7 stack.append(30)
8 print("Stack setelah push:",stack)
9
10 #pop elemen dari stack
11 elemen=stack.pop()
12 print("elemen yang di-pop:",elemen)
13 print("Stack setelah pop:",stack)
14
15 #peek elemen teratas
16 if stack:
17     print("Elemen teratas:",stack[-1])
18 else:
19     print("Stack kosong")
```

Gambar 1. output halo py.

1. implementasi stack dalam python - Ini adalah komentar yang menunjukkan bahwa kode di bawahnya akan mengimplementasikan stack dalam Python.
2. `stack = []` - Ini mendeklarasikan sebuah variabel stack yang berupa list kosong.
- 3-7. Ini adalah baris-baris kode yang memanipulasi stack dengan menggunakan metode `append()` untuk menambahkan elemen ke stack.
8. `print("Stack setelah push:", stack)` - Ini akan mencetak stack setelah elemen-elemen ditambahkan.
10. `#pop elemen dari stack` - Ini adalah komentar yang menjelaskan bahwa kode di bawahnya akan menghapus elemen dari stack.
11. `elemen = stack.pop()` - Ini akan menghapus dan menyimpan elemen terakhir yang ditambahkan ke stack.

12. `print("Elemen yang di-pop:", elemen)` - Ini akan mencetak elemen yang baru saja dihapus dari stack.

13. `print("Stack setelah pop:", stack)` - Ini akan mencetak stack setelah elemen dihapus.

15. `#peek elemen teratas` - Ini adalah komentar yang menjelaskan bahwa kode di bawahnya akan melihat elemen teratas stack.

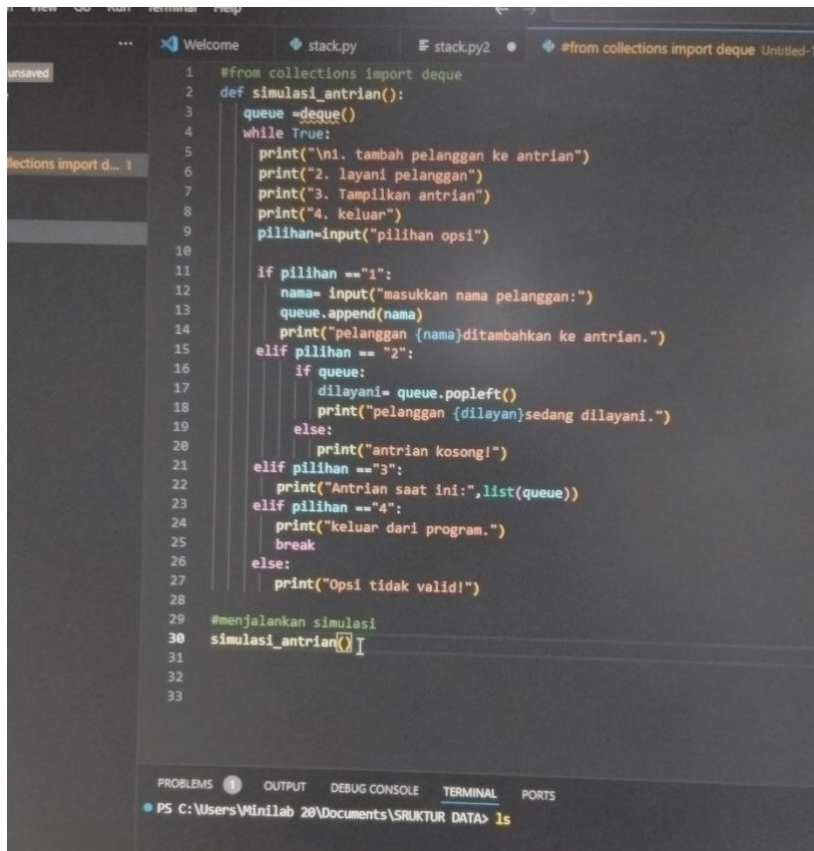
16. `if stack:` - Ini adalah pemeriksaan apakah stack masih berisi elemen.

17. `print("Elemen teratas:", stack[-1])` - Ini akan mencetak elemen teratas stack tanpa menghapusnya.

18. `else:` - Ini adalah bagian else dari if statement di atas.

19. `print("Stack kosong")` - Ini akan mencetak pesan jika stack kosong.

Secara keseluruhan, kode ini mendemonstrasikan beberapa operasi dasar pada stack, seperti push, pop, dan peek.

A screenshot of a Python IDE window. The main editor shows a file named 'stack.py2' with a Python script. The script imports 'deque' from 'collections' and defines a function 'simulasi\_antrian()'. Inside the function, a 'queue' is initialized as a 'deque()'. A 'while True' loop handles menu options: '1' to add a customer, '2' to serve a customer, '3' to display the queue, and '4' to exit. The script also includes a call to 'simulasi\_antrian()' at the bottom. The bottom of the IDE shows a terminal window with the command 'ls' and the directory path 'C:\Users\Winilab 20\Documents\SRUKTUR DATA>'.

```
1 #from collections import deque
2 def simulasi_antrian():
3     queue = deque()
4     while True:
5         print("\n1. tambah pelanggan ke antrian")
6         print("2. layani pelanggan")
7         print("3. Tampilkan antrian")
8         print("4. keluar")
9         pilihan = input("pilihan opsi")
10
11     if pilihan == "1":
12         nama = input("masukkan nama pelanggan:")
13         queue.append(nama)
14         print("pelanggan {nama} ditambahkan ke antrian.")
15     elif pilihan == "2":
16         if queue:
17             dilayani = queue.popleft()
18             print("pelanggan {dilayan} sedang dilayani.")
19         else:
20             print("antrian kosong!")
21     elif pilihan == "3":
22         print("Antrian saat ini:", list(queue))
23     elif pilihan == "4":
24         print("keluar dari program.")
25         break
26     else:
27         print("Ops! tidak valid!")
28
29 #menjalankan simulasi
30 simulasi_antrian()
31
32
33
```

**Gambar 1.** Ouput file halo.py

Baik, saya akan menjelaskan baris per baris dari kode tersebut:

1. "#from collections import deque Untitled-1 1" adalah komentar yang menunjukkan bahwa kode ini mengimpor modul "deque" dari paket "collections".
2. "def simulasi\_antrian():" mendefinisikan sebuah fungsi bernama "simulasi\_antrian" yang akan dijalankan.
3. "queue = deque()" membuat sebuah antrian kosong menggunakan struktur data "deque" dari modul "collections".
4. "while True:" memulai sebuah loop tak terbatas.
5. 'print("1. tambah pelanggan ke antrian")' menampilkan teks "1. tambah pelanggan ke antrian" ke output.
6. 'print("2. layani pelanggan")' menampilkan teks "2. layani pelanggan" ke output.
7. 'print("3. tampilkan antrian")' menampilkan teks "3. tampilkan antrian" ke output.

8. `'print("4. keluar")'` menampilkan teks "4. keluar" ke output.
9. `"pilihan-input('pilihan opsi')"` meminta pengguna untuk memasukkan input yang akan disimpan dalam variabel "pilihan".
10. `"if pilihan == "1":"` memulai sebuah kondisi if untuk memeriksa apakah nilai "pilihan" adalah "1".
11. `"nama = input("masukkan nama pelanggan:")"` meminta pengguna untuk memasukkan nama pelanggan yang akan disimpan dalam variabel "nama".
12. `"queue.append(nama)"` menambahkan "nama" ke dalam antrian.
13. `'print("pelanggan {nama}ditambahkan ke antrian.")'` menampilkan pesan yang menunjukkan bahwa pelanggan dengan nama tertentu telah ditambahkan ke antrian.
14. `"elif pilihan == "2":"` memulai sebuah kondisi elif untuk memeriksa apakah nilai "pilihan" adalah "2".
15. `"if queue:"` memeriksa apakah antrian tidak kosong.
16. `"dilayanin queue.popleft()"` mengeluarkan dan menyimpan elemen pertama dari antrian.
17. `'print("pelanggan {dilayanin}sedang dilayani.")'` menampilkan pesan yang menunjukkan bahwa pelanggan tertentu sedang dilayani.
18. `"else:"` memulai sebuah kondisi else yang akan dijalankan jika kondisi sebelumnya tidak terpenuhi.
19. `'print("antrian kosong!")'` menampilkan pesan bahwa antrian kosong.
20. `"elif pilihan == "3":"` memulai sebuah kondisi elif untuk memeriksa apakah nilai "pilihan" adalah "3".
21. `'print("Antrian saat ini:",list(queue))'` menampilkan daftar elemen yang ada dalam antrian saat ini.
22. `"elif pilihan == "4":"` memulai sebuah kondisi elif untuk memeriksa apakah nilai "pilihan" adalah "4".
23. `'print("Keluar dari program.")'` menampilkan pesan bahwa pengguna keluar dari program.
24. `"break"` keluar dari loop tak terbatas.
25. `"else:"` memulai sebuah kondisi else yang akan dijalankan jika kondisi sebelumnya tidak terpenuhi.



26. `'print("Opsi tidak valid!")'` menampilkan pesan bahwa opsi yang dipilih tidak valid.
27. `"#menjalankan simulasi"` adalah komentar yang menunjukkan bahwa kode selanjutnya akan menjalankan simulasi antrian.
28. `"simulasi_antrian()"` memanggil fungsi `"simulasi_antrian"` yang telah didefinisikan sebelumnya.

## E. Daftar Pustaka

"Introduction to Algorithms" by Thomas H. Cormen, Charles B. Leiserson, and Ronald L. Rivest

Buku ini memberikan pengantar komprehensif tentang algoritma dan struktur data, termasuk stack dan queue.

"Data Structures and Algorithms in Python" by Michael T. Goodrich, et al.

Buku ini fokus pada implementasi struktur data dan algoritma dalam bahasa Python, dengan penjelasan yang jelas tentang stack dan queue.

"The Art of Computer Programming" by Donald E. Knuth

Buku ini adalah karya klasik yang membahas berbagai aspek pemrograman komputer, termasuk desain dan analisis algoritma, di mana stack dan queue dibahas secara mendalam.

"Operating System Concepts" by Abraham S. Tanenbaum and H. M. Bos

Buku ini menjelaskan konsep-konsep dasar sistem operasi, termasuk manajemen proses dan penggunaan stack dan queue dalam konteks ini.

"Introduction to Computer Science using Java" by Y. Daniel Lieberman and C. H. Papadimitriou

Buku ini menggunakan bahasa Java untuk mengajarkan konsep-konsep dasar dalam ilmu komputer, termasuk implementasi stack dan queue.

Dengan memahami dasar teori stack dan queue serta referensi-referensi di atas, Anda dapat lebih mendalami konsep-konsep ini dan menerapkannya dalam berbagai konteks pemrograman dan sistem komputer.