LAPORAN PRAKTIKUM
ALGORITMA & STRUKTUR DATA
MODUL 2

STACK & QUEUE

Oleh:

Rika Fauliana Rahmi NIM. 2410817120017

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
APRIL 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM ALGORITMA & STRUKTUR DATA
MODUL 2

Laporan Praktikum Algoritma & Struktur Data Modul 2: Stack & Queue ini disusun sebagai syarat lulus mata kuliah Praktikum Algoritma & Struktur Data. Laporan Prakitkum ini dikerjakan oleh:

Nama Praktikan : Rika Fauliana Rahmi
NIM : 2410817120017

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Muhammad Fauzan Ahsani NIM. 2310817310009	Muti'a Maulida, S.Kom., M.TI. NIP. 198810272019032013

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

1. Apa Perbedaan Stack dan Queue?

A. Jawaban Soal

Stack :

Prinsip dasar: LIFO (Last In, First Out), yang berarti elemen terakhir yang dimasukkan adalah elemen pertama yang akan dikeluarkan.

Operasi utama:

- Push: Menambahkan elemen ke bagian atas stack.
- Pop: Menghapus elemen dari bagian atas stack.

Contoh aplikasi: Pengolahan bahasa pemrograman (misalnya, untuk undo/redo), penelusuran (searching) dalam algoritma seperti DFS (Depth-First Search), dll.

Queue :

Prinsip dasar: FIFO (First In, First Out), yang berarti elemen pertama yang dimasukkan adalah elemen pertama yang akan dikeluarkan.

Operasi utama:

- Enqueue: Menambahkan elemen ke belakang antrian.
- Dequeue: Menghapus elemen dari depan antrian.

Contoh aplikasi: Manajemen proses dalam sistem operasi (misalnya, penjadwalan proses), antrian dalam sistem jaringan, dll.

SOAL 2

2. Cobalah contoh program berikut, running dan analisis hasilnya!

Tabel 1 Tabel Source Code Soal 2

```
1  int penuh()
2  {
3          if(Tumpuk.atas == MAX -1)
4              return 1;
5          else
6              return 0;
7  }
8
9  void input (int data)
10 {
11         if(kosong()==1)
12         {
13             Tumpuk.atas++;
14             Tumpuk.data[Tumpuk.atas] = data;
15             cout <<"Data"<< Tumpuk.data[Tumpuk.atas]
16                <<"Masuk Ke Stack";
17         }
18         else if(penuh()==0)
19         {
20             Tumpuk.atas++;
21             Tumpuk.data[Tumpuk.atas] = data;
22             cout <<"Data"<< Tumpuk.data[Tumpuk.atas]
23                <<"Masuk Ke Stack";
24         }
25         else
26             cout <<"Tumpukan Penuh";
27         }
28
29 void hapus()
30 {
31         if(kosong()==0)
32         {
33             cout <<"Data Teratas Sudah Terambil";
34             Tumpuk.atas--;
35         }
36         else
37             cout <<"Data Kosong";
38         }
39
40
```

```
41 void tampil()
42 {
43         if(kosong()==0)
44         {
45             for(int i=Tumpuk.atas; i>=0; i--)
46             {
47
48                cout <<"\nTumpukan Ke "<<i<<"="
49                <<Tumpuk.data[i];
50             }
51         }
52         else
53             cout <<"Data Kosong";
54         }
55
56 void bersih()
57 {
58         Tumpuk.atas = -1;
59         cout <<"Tumpukan Kosong!";
60 }
```

A. Source Code

Perbaikan dari source code yang diberikan :

```
// write the definition (body) of the stack here using struct and array

// the element of the stack are integers

// write the function to implement all the operation needed

// push, pop, isEmpty, isFull, display, reset, init

// write the menu function in main scope

// to insert, delete, display, reset, exit
```

Tabel 2 Tabel Source Code Jawaban Soal 2

```
1  #include <iostream>
2  #include <conio.h>
3  using namespace std;
4
```

```
5   #define MAX 10
6
7   struct Stack {
8       int data[MAX];
9       int atas;
10  };
11
12  Stack Tumpuk;
13
14  void inisialisasi() {
15      Tumpuk.atas = -1;
16  }
17
18  int kosong() {
19      return Tumpuk.atas == -1;
20  }
21
22  int penuh() {
23      return Tumpuk.atas == MAX - 1;
24  }
25
26  void input(int data) {
27      if (!penuh()) {
28          Tumpuk.atas++;
29          Tumpuk.data[Tumpuk.atas] = data;
30         cout << "Data " << data << " Masuk ke Stack\n";
31      } else {
32          cout << "Tumpukan Penuh\n";
33      }
34  }
35
36  void hapus() {
37      if (!kosong()) {
38        cout << "Data " << Tumpuk.data[Tumpuk.atas] <<
    " Teratas Sudah Terambil\n";
39          Tumpuk.atas--;
40      } else {
41          cout << "Data Kosong\n";
42      }
43  }
44
45  void tampil() {
46      if (!kosong()) {
47          cout << "Isi Stack:\n";
48          for (int i = Tumpuk.atas; i >= 0; i--) {
```

```
49              cout << "Tumpukan ke " << i << " = " << Tumpuk.data[i] << endl;
50          }
51      } else {
52          cout << "Data Kosong\n";
53      }
54  }
55
56  void bersih() {
57      Tumpuk.atas = -1;
58      cout << "Tumpukan Kosong!\n";
59  }
60
61  int main() {
62      int pilihan, nilai;
63      inisialisasi();
64
65      do {
66          cout << "\nSTACK\n";
67          cout << "==============\n";
68          cout << "1. PUSH\n";
69          cout << "2. POP\n";
70          cout << "3. CETAK STACK\n";
71          cout << "4. BERSIHKAN STACK\n";
72          cout << "5. QUIT\n";
73          cout << "PILIHAN : ";
74          cin >> pilihan;
75
76          switch (pilihan) {
77              case 1:
78                  cout << "Masukkan Nilai: ";
79                  cin >> nilai;
80                  input(nilai);
81                  break;
82              case 2:
83                  hapus();
84                  break;
85              case 3:
86                  tampil();
87                  break;
88              case 4:
89                  bersih();
90                  break;
91              case 5:
92                  cout << "Program Selesai.\n";
```

```
93                  break;
94              default:
95                  cout << "Pilihan tidak valid!\n";
96          }
97             cout << "Tekan tombol apa saja untuk
    melanjutkan...\n";
98          getch();
99      } while (pilihan != 5);
100
101     return 0;
102 }
```

B. Output Program

Gambar 1 Memasukkan nilai ke dalam stack

Gambar 2 Menampilkan isi stack

Gambar 3 Melakukan pop pada stack

```
STACK
==============
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 3
Isi Stack:
Tumpukan ke 3 = 4
Tumpukan ke 2 = 3
Tumpukan ke 1 = 2
Tumpukan ke 0 = 1
Tekan tombol apa saja untuk melanjutkan...
```

Gambar 4 Tampilan setelah nilai teratas di pop

```
STACK
==============
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 4
Tumpukan Kosong!
Tekan tombol apa saja untuk melanjutkan...
```

Gambar 5 Membersihkan stack

```
STACK
==============
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 1
Masukkan Nilai: 11
Tumpukan Penuh
Tekan tombol apa saja untuk melanjutkan...
```

Gambar 6 Tampilan saat stack penuh

```
STACK
==============
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 5
Program Selesai.
Tekan tombol apa saja untuk melanjutkan...
PS C:\Users\Lenovo\OneDrive\Documents>
```

Gambar 7 Tampilan saat program stack selesai

C. Pembahasan

Pada baris [1], `#include <iostream>` digunakan untuk input-output seperti cin dan cout.

Pada baris [2], `#include <conio.h>` digunakan untuk menggunakan `getch()` (fungsi untuk menunggu input karakter dari keyboard tanpa menekan enter).

Pada baris [3], `using namespace std;:` digunakan agar tidak perlu menulis `std::`.

Pada baris [5], baris ini mendefinisikan nilai maksimum elemen dalam stack, yaitu 10.

Pada baris [7]-[10], stack adalah struct dengan `data[MAX]` array untuk menyimpan elemen stack. `atas:` menyimpan indeks elemen paling atas dalam stack.

Pada baris [12], baris ini membuat variabel global `Tumpuk` dari tipe `Stack.`

Pada baris [14]-[16], baris ini mengatur nilai awal stack kosong (`atas = -1` menandakan belum ada elemen).

Pada baris [18]-[20], baris ini mengembalikan `true (1)` jika stack kosong.

Pada baris [22]-[24], baris ini mengembalikan `true (1)` jika stack sudah mencapai batas maksimum.

Pada baris [26]-[34], baris ini mengecek apakah stack tidak penuh. Jika tidak penuh: Increment `atas`, lalu masukkan data ke `Tumpuk.data[atas]`. Jika penuh, tampilkan pesan.

Pada baris [36]-[43], baris ini mengecek apakah stack tidak kosong. Jika tidak kosong: Tampilkan data paling atas, turunkan `atas` untuk menghapusnya dari stack. Jika kosong, tampilkan pesan.

Pada baris [45]-[54], baris ini mengecek apakah stack tidak kosong. Jika tidak kosong, cetak isi stack dari atas ke bawah (sesuai prinsip stack). Jika kosong, tampilkan pesan.

Pada baris [56]-[59], baris ini mengatur `atas` kembali ke -1, artinya stack dikosongkan.

Pada baris [61]-[63], baris ini mendeklarasikan variabel. Memanggil `inisialisasi()` untuk memastikan stack kosong saat program mulai.

Pada baris [65]-[74], baris ini menampilkan menu pilihan berulang kali hingga user memilih 5.
Pada baris [76]-[96], baris ini menjalankan fungsi sesuai pilihan user.
Pada baris [97]-[102], Setelah menjalankan perintah, program menunggu input tombol `(getch())`. Ulangi sampai user memilih keluar (`5`).

Tambahan :

`inisialisasi()`, mengatur stack dalam kondisi kosong (`atas = -1`)

`kosong()`, mengecek apakah stack kosong

`penuh()`, mengecek apakah stack penuh

`input(data)`, menambahkan elemen ke dalam stack (Push)

`hapus()`, menghapus elemen teratas dari stack (Pop)

`tampil()`, menampilkan isi stack dari atas ke bawah

`bersih()`, ngosongkan seluruh isi stack (reset `atas = -1`)

SOAL 3

3. Cobalah contoh program berikut, running dan analisis hasilnya!

Tabel 3 Tabel Source Code Soal 3

```
 1  #include <iostream>
 2  #include <conio.h>
 3  #include <stdlib.h>
 4  #define n 20
 5  using namespace std;
 6  void INSERT();
 7  void DELETE();
 8  void CETAKLAYAR();
 9  void Inisialisasi();
10  void RESET();
11  int PIL, F, R;
12  char PILIHAN[1], HURUF;
13  char Q[n];
14  int main()
15  {
16          Inisialisasi();
17          do
18          {
19             cout << "QUEUE" << endl;
20             cout << "==============" << endl;
21             cout << "1. INSERT" << endl;
22             cout << "2. DELETE" << endl;
23             cout << "3. CETAK QUEUE" << endl;
24             cout << "4. QUIT" << endl;
25             cout << "PILIHAN ANDA : "; cin >>
    PILIHAN;
26             PIL = atoi(PILIHAN);
27             switch (PIL)
28             {
29             case 1:
30               INSERT();
31               break;
32             case 2:
33               DELETE();
34               break;
35             case 3:
36               CETAKLAYAR();
37               break;
38             default:
39               cout << "TERIMA KASIH" << endl;
```

```
40              break;
41            }
42            cout << "Press any key to continue" <<
    endl;
43            getch();
44            system("cls");
45          }
46          while (PIL < 4);
```

A. Source Code

Perbaikan dari source code yang diberikan :

```
// write the definition (body) of the queue here using
struct and array

// the element of the queue are char

// write the function to implement all the operation needed

// enqueue, dequeue, isEmpty, isFull, display, reset, init

// write the menu function in main scope

// to insert, delete, display, reset, exit
```

Tabel 4 Tabel Source Code Jawaban Soal 3

```
1   #include <iostream>
2   #include <conio.h>
3   #include <stdlib.h>
4   #define n 20
5   using namespace std;
6
7   struct Queue {
8       char data[n];
9       int front;
10      int rear;
11  };
12
13  Queue q;
14
15  void Inisialisasi() {
16      q.front = -1;
17      q.rear = -1;
```

```
18  }
19
20  bool isFull() {
21      return q.rear == n - 1;
22  }
23
24  bool isEmpty() {
25      return q.front == -1 || q.front > q.rear;
26  }
27
28  void INSERT() {
29      char huruf;
30      if (isFull()) {
31          cout << "Queue Penuh!" << endl;
32          return;
33      }
34
35      cout << "Masukkan Huruf: ";
36      cin >> huruf;
37
38      if (isEmpty()) {
39          q.front = q.rear = 0;
40      } else {
41          q.rear++;
42      }
43      q.data[q.rear] = huruf;
44       cout << "Data: " << huruf << " Masuk ke dalam
    Queue" << endl;
45  }
46
47  void DELETE() {
48      if (isEmpty()) {
49          cout << "Queue Kosong!" << endl;
50          return;
51      }
52      cout << "Data: " << q.data[q.front] << " Dihapus
    dari Queue" << endl;
53      q.front++;
54  }
55
56  void CETAKLAYAR() {
57      if (isEmpty()) {
58          cout << "Queue Kosong!" << endl;
59          return;
60      }
```

```
61
62      cout << "Isi Queue:\n";
63      for (int i = q.front; i <= q.rear; i++) {
64          cout << "Queue ke " << i << " = " << q.data[i]
    << endl;
65      }
66  }
67
68  void RESET() {
69      q.front = q.rear = -1;
70      cout << "Queue telah di-reset.\n";
71  }
72
73  int main() {
74      int PIL;
75      char PILIHAN[1];
76
77      Inisialisasi();
78
79      do {
80          cout << "QUEUE" << endl;
81          cout << "==============" << endl;
82          cout << "1. INSERT" << endl;
83          cout << "2. DELETE" << endl;
84          cout << "3. CETAK QUEUE" << endl;
85          cout << "4. RESET QUEUE" << endl;
86          cout << "5. QUIT" << endl;
87          cout << "PILIHAN ANDA : ";
88          cin >> PILIHAN;
89
90          PIL = atoi(PILIHAN); // Ubah dari string ke
    int
91
92          switch (PIL) {
93              case 1:
94                  INSERT();
95                  break;
96              case 2:
97                  DELETE();
98                  break;
99              case 3:
100                 CETAKLAYAR();
101                 break;
102             case 4:
103                 RESET();
```

```
104                     break;
105                 default:
106                     cout << "TERIMA KASIH" << endl;
107                     break;
108             }
109
110                cout << "Tekan tombol apa saja untuk
    melanjutkan..." << endl;
111             getch();
112             system("cls");
113
114         } while (PIL < 4);
115
116         return 0;
117 }
```

B. Output Program

```
QUEUE
==============
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 1
Masukkan Huruf: A
Data: A Masuk ke dalam Queue
Tekan tombol apa saja untuk melanjutkan...
```

Gambar 8 Memasukkan nilai ke dalam queue

```
QUEUE
==============
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 3
Queue ke 0 = A
Queue ke 1 = B
Queue ke 2 = C
Queue ke 3 = D
Queue ke 4 = E
Tekan tombol apa saja untuk melanjutkan...
```

Gambar 9 Menampilkan cetak queue

```
QUEUE
==============
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 2
Data: A Dihapus dari Queue
Tekan tombol apa saja untuk melanjutkan...
```

Gambar 10 Menghapus huruf yang sudah dimasukkan di queue

```
QUEUE
==============
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 3
Queue ke 1 = B
Queue ke 2 = C
Queue ke 3 = D
Queue ke 4 = E
Tekan tombol apa saja untuk melanjutkan...
```

Gambar 11 Tampilan setelah melakukan delete

```
QUEUE
==============
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 4
Queue telah di-reset.
Tekan tombol apa saja untuk melanjutkan...
```

Gambar 12 Tampilan saat queue di reset

```
QUEUE
==============
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 1
Masukkan Huruf: u
Queue Penuh!
Tekan tombol apa saja untuk melanjutkan...
```

Gambar 13 Tampilan saat queue penuh

Gambar 14 Tampilan saat queue di quit

C. Pembahasan

Pada baris [1], `#include <iostream>` digunakan untuk input-output seperti cin dan cout.

Pada baris [2], `#include <conio.h>` digunakan untuk menggunakan `getch()` (fungsi untuk menunggu input karakter dari keyboard tanpa menekan enter).

Pada baris [3], `#include <stdlib.h>` digunakan untuk fungsi `atoi()` (konversi string ke integer).

Pada baris [4], `#define n 20` mendefinisikan ukuran maksimum queue menjadi 20 elemen.

Pada baris [5], `using namespace std;:` digunakan agar tidak perlu menulis `std::`.

Pada baris [7]-[11], `data[n]` array untuk menyimpan elemen queue bertipe `char`. `front` indeks elemen paling depan. `rear` indeks elemen paling belakang.

Pada baris [13], baris ini membuat variabel global `q` bertipe `Queue`.

Pada baris [15]-[18], baris ini mengatur `front` dan `rear` ke `-1` sebagai tanda queue kosong.

Pada baris [20]-[22], baris ini engembalikan `true` jika indeks `rear` sudah mencapai akhir array.

Pada baris [24]-[26], `true` jika belum ada data (`front == -1`) atau semua data sudah dikeluarkan (`front > rear`).

Pada baris [28]-[33], baris ini mengecek apakah queue penuh, jika iya, tampilkan pesan.

Pada baris [35]-[36], baris ini meminta input karakter dari user.

Pada baris [38]-[42], baris ini jika kosong, maka `front` dan `rear` di-set ke 0. Jika tidak, `rear` dinaikkan untuk posisi selanjutnya.

Pada baris [43]-[45], baris ini meyimpan huruf ke array, dan tampilkan pesan sukses.

Pada baris [47]-[51], baris ini mengecek apakah queue kosong.

Pada baris [52]-[54], baris ini menampilkan elemen di `front`, lalu `front` dinaikkan (elemen "keluar").

Pada baris [56]-[60], baris ini mengecek apakah queue kosong.

Pada baris [62]-[66], baris ini menampilkan isi queue dari `front` hingga `rear`.

Pada baris [68]-[71], baris ini mengembalikan kondisi queue ke kosong, seperti awal program.

Pada baris [73]-[75], `PIL` menyimpan pilihan menu. `PILIHAN` digunakan agar input tetap berupa string, lalu diubah ke int pakai `atoi`.

Pada baris [77], inisialisasi queue sebelum digunakan.

Pada baris [79]-[90], baris ini menampilkan menu dan meminta pilihan dari user.

Pada baris [92]-[108], baris ini menyesuaikan fungsi yang dipanggil berdasarkan pilihan user.

Pada baris [110]-[114], baris ini menunggu tombol apa saja, lalu membersihkan layar. loop berjalan hingga `PIL >= 4`.

Pada baris [116]-[117], baris ini menandakan program selesai.

Tambahan :

`inisialisasi()`, mengatur stack dalam kondisi kosong (`atas = -1`)

`isfull()`, mengembalikan `true` jika queue sudah penuh (`rear == n - 1`).

`isEmpty()`, Mengembalikan `true` jika queue kosong (`front == -1 atau front > rear`).

`INSERT()`, Memasukkan karakter ke akhir queue. Menyesuaikan indeks `rear`.

`DELETE()`, menghapus karakter dari depan queue. Menyesuaikan indeks `front`.

`CETAKLAYAR()`, menampilkan isi queue dari posisi `front` sampai `rear`.

`RESET()`, mengosongkan queue dengan mengatur `front` dan `rear` kembali ke `-1`.

`system("cls")`, membersihkan layar.

`getch()`, menunggu input tombol apapun untuk pause sebelum membersihkan layar.

`atoi(PILIHAN)`, Mengubah input string ke integer agar bisa digunakan di `switch`.

Tautan Git

https://github.com/DSA25-ULM/task-2-stack-and-queue-rikafaulianarahmi