LAPORAN PRAKTIKUM
ALGORITMA & STRUKTUR DATA
MODUL 2

STACK AND QUEUE

Oleh:

Indra Suryadilaga NIM. 2410817310014

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

Nama Praktikan : Indra Suryadilaga
NIM : 2410817310014

Menyetujui,	Mengetahui,
Asisten Praktikum	Dosen Penanggung Jawab Praktikum
Zulfa Auliya Akbar	Andreyan Rizky Baskara, S.Kom., M.Kom
NIM. 2210817210010	NIP. 199307032019031011

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

Apa perbedaan Stack dengan Queue?

Pembahasan

Stack dan Queue merupakan dua jenis struktur data linear yang memiliki perbedaan utama dalam cara menyimpan dan mengambil data.

Stack bekerja dengan prinsip Last In, First Out (LIFO), artinya data yang terakhir dimasukkan akan menjadi data pertama yang dikeluarkan. Struktur ini dapat diibaratkan seperti tumpukan buku, di mana buku yang paling atas akan diambil terlebih dahulu. Operasi dasar pada Stack meliputi push (menambahkan data ke atas tumpukan) dan pop (menghapus data dari atas tumpukan).

Sebaliknya, Queue menggunakan prinsip First In, First Out (FIFO), yang berarti data yang pertama kali dimasukkan akan menjadi data pertama yang dikeluarkan. Queue dapat dianalogikan seperti antrean di loket, di mana orang yang datang lebih dahulu akan dilayani lebih dahulu. Operasi dasar pada Queue terdiri dari enqueue (menambahkan data ke belakang antrean) dan dequeue (menghapus data dari depan antrean).

Kedua struktur data ini sangat penting dalam berbagai aplikasi komputer, seperti manajemen memori, pemrosesan data, dan pengendalian antrian.

SOAL 2

A. Source Code

```
#include <iostream>
#include <conio.h>
#include <string>
using namespace std;

const int max_stack = 10;
string PILIHAN, data;
int PIL;

struct Stack {
    int atas;
    string data[max_stack];
} Tumpuk;

int penuh() {
    if (Tumpuk.atas == max_stack - 1)
        return 1;
    else
        return 0;
}

int kosong() {
    if (Tumpuk.atas == -1)
        return 1;
    else
        return 0;
}

void input()
{
    if (penuh() == 0){
        cout << "Masukkan data: ";
        cin >> data;
        Tumpuk.atas++;
        Tumpuk.data[Tumpuk.atas] = data;
        cout << "Data " << Tumpuk.data[Tumpuk.atas]
             << " Masuk Ke Stack" << endl;
    }
    else
        cout << "Tumpukan Penuh" << endl;
}


void hapus() {
    if (kosong() == 0) {
        cout << "Data \"" << Tumpuk.data[Tumpuk.atas] << "\" teratas sudah terambil\n";
        Tumpuk.atas--;
    }
    else {
        cout << "Data kosong\n";
    }
}
```

Gambar 1. Source Code Soal 2

```
54  void tampil() {
55      if (kosong() == 0) {
56          for (int i = Tumpuk.atas; i >= 0; i--) {
57              cout << "Tumpukan ke-" << i << " = \"" << Tumpuk.data[i] << "\"" << endl;
58          }
59      }
60      else {
61          cout << "Tumpukan kosong\n";
62      }
63  }
64
65  void bersih() {
66      Tumpuk.atas = -1;
67      cout << "Tumpukan kosong!\n";
68  }
69
70  int main() {
71      Tumpuk.atas = -1;
72      do {
73          cout << "STACK" << endl;
74          cout << "===============" << endl;
75          cout << "1. Input" << endl;
76          cout << "2. Hapus" << endl;
77          cout << "3. Cetak stack" << endl;
78          cout << "4. Keluar" << endl;
79          cout << "Pilihan : ";
80          cin >> PILIHAN;
81          PIL = stoi(PILIHAN);
82
83          switch (PIL) {
84              case 1:
85                  input();
86                  break;
87              case 2:
88                  hapus();
89                  break;
90              case 3:
91                  tampil();
92                  break;
93              case 4:
94                  cout << "TERIMA KASIH" << endl;
95                  break;
96              default:
97                  cout << "Pilihan tidak valid" << endl;
98                  break;
99          }
100         cout << "Press any key to continue..." << endl;
101         getch();
102         system("cls");
103     } while (PIL != 4);
104     return 0;
105 }
106
```

Gambar 2. Source Code Soal 2

B. Pembahasan

1. Penambahan Library dan Namespace

   Bagian ini mendeklarasikan library yang dibutuhkan oleh program:

   a. <iostream>: Diperlukan untuk operasi input/output standar seperti cin dan cout

b. <conio.h>: Library yang menyediakan fungsi getch() untuk menahan layar sampai pengguna menekan tombol
c. <string>: Library yang menambahkan dukungan untuk manipulasi string dalam C++.

Penggunaan using namespace std; memungkinkan program menggunakan fungsi-fungsi dari namespace standard (std) tanpa harus menuliskan prefiks "std::" setiap kali, seperti std::cout atau std::string.

Table 1. Source Code Pembahasan Soal 2

```
#include <iostream>
#include <conio.h>
#include <string>
using namespace std;
```

2. Deklarasi Variabel global

Bagian ini mendefinisikan konstanta dan variabel global yang digunakan dalam program:

a. const int max_stack = 10;: Mendefinisikan ukuran maksimum stack sebagai konstanta bernilai 10.
b. string PILIHAN;: Variabel untuk menyimpan input pilihan menu dari pengguna dalam bentuk string.
c. int PIL;: Variabel untuk menyimpan pilihan menu yang sudah dikonversi ke integer untuk digunakan dalam struktur kontrol program.
d. string data;: Variabel untuk menyimpan data yang akan dimasukkan ke dalam stack.

Table 2. Source Code Pembahasan Soal 2

```
const int max_stack = 10;
string PILIHAN;
int PIL;
string data;
```

3. Struktur Data Stack

Bagian ini mendefinisikan struktur data stack dan variabel instance-nya:

a. struct Stack: Mendefinisikan tipe data struktur bernama Stack yang terdiri dari:

1) int atas: Variabel yang berfungsi sebagai pointer/indeks untuk menunjukkan posisi elemen teratas pada stack
2) string data[max_stack]: Array dengan ukuran max_stack untuk menyimpan elemen-elemen stack dalam bentuk string

b. Tumpuk: Variabel instance dari struktur Stack yang akan digunakan untuk operasi stack dalam program.

Table 3. Source Code Pembahasan Soal 2

```
struct Stack {
    int atas;
    string data[max_stack];
} Tumpuk;
```

4. Fungsi penuh()

Fungsi penuh() bertugas untuk memeriksa apakah stack dalam keadaan penuh. Fungsi ini mengembalikan nilai:

c. 1 (true) jika stack penuh.
d. 0 (false) jika stack tidak penuh.

Logika yang digunakan adalah dengan memeriksa apakah indeks Tumpuk.atas sudah mencapai nilai maksimum stack dikurangi 1 (max - 1).

Table 4. Source Code Pembahasan Soal 2

```
int penuh()
{
    if (Tumpuk.atas == max - 1)
        return 1;
    else
        return 0;
}
```

5. Fungsi kosong()

Fungsi kosong() bertugas untuk memeriksa apakah stack dalam keadaan kosong. Fungsi ini mengembalikan nilai:

a. 1 (true) jika stack kosong.
b. 0 (false) jika stack tidak kosong.

Logika yang digunakan adalah dengan memeriksa apakah indeks Tumpuk.atas sudah mencapai nilai maksimum stack dikurangi 1 (max - 1).

Table 5. Source Code Pembahasan Soal 2

```
int kosong() {
    if (Tumpuk.atas == -1)
        return 1;
    else
        return 0;
}
```

6. Fungsi input()

Fungsi input() digunakan untuk memasukkan data ke dalam stack. Fungsi ini memiliki beberapa kondisi:

a. Jika stack tidak penuh (penuh() == 0), maka data di input lalu nilai atas ditambah 1 dan data dimasukkan.
b. Jika stack tidak penuh (penuh() == 0), maka data juga di input lalu nilai atas ditambah 1 dan data dimasukkan.
c. Jika kondisi lainnya (yaitu stack penuh), maka ditampilkan pesan "Tumpukan Penuh".

Logika yang digunakan adalah dengan memeriksa apakah nilai dari fungsi penuh bernilai false.

Table 6. Source Code Pembahasan Soal 2

```
void input(){
    if (penuh() == 0){
        cout << "Masukkan data: ";
        cin >> data;
        Tumpuk.atas++;
        Tumpuk.data[Tumpuk.atas] = data;
        cout << "Data " << Tumpuk.data[Tumpuk.atas] << "
Masuk Ke Stack" << endl;
    }else if (penuh() == 0) {
        cout << "Masukkan data: ";
        cin >> data;
        Tumpuk.atas++;
        Tumpuk.data[Tumpuk.atas] = data;
        cout << "Data " << Tumpuk.data[Tumpuk.atas] << "
masuk ke stack\n";
    }
    else
        cout << "Tumpukan Penuh" << endl;
```

```
}
```

7. Fungsi hapus()

   Fungsi hapus() bertugas untuk menghapus elemen teratas dari stack. Fungsi ini:

   a. Jika tidak kosong, maka indeks pointer Tumpuk.atas dikurangi 1, yang secara efektif menghapus elemen teratas.
   b. Jika kosong, maka ditampilkan pesan "Data Kosong".

   Fungsi ini tidak benar-benar menghapus data secara fisik dari array, melainkan hanya mengubah indeks teratas yang dapat diakses.

Table 7. Source Code Pembahasan Soal 2

```
void hapus() {
    if (kosong() == 0) {
        cout << "Data \"" << Tumpuk.data[Tumpuk.atas] << "\"
teratas sudah terambil\n";
        Tumpuk.atas--;
    }
    else {
        cout << "Data kosong\n";
    }
}
```

8. Fungsi tampil ()

   Fungsi tampil() bertugas untuk menampilkan seluruh isi stack dari atas ke bawah. Fungsi ini:

   a. Jika tidak kosong, menampilkan seluruh elemen stack mulai dari indeks teratas hingga indeks terbawah (0) .
   b. Jika kosong, menampilkan pesan "Tumpukan Kosong".

   Implementasi ini sesuai dengan prinsip LIFO (Last In First Out) dari stack, di mana elemen teratas ditampilkan terlebih dahulu.

Table 8. Source Code Pembahasan Soal 2

```
void tampil() {
    if (kosong() == 0) {
        for (int i = Tumpuk.atas; i >= 0; i--) {
```

```
            cout << "Tumpukan ke-" << i << " = \"" <<
Tumpuk.data[i] << "\"" << endl;
        }
    }
    else {
        cout << "Tumpukan kosong\n";
    }
}
```

9. Fungsi bersih()

Fungsi bersih() bertugas untuk mengosongkan stack. Fungsi ini:

a. Menetapkan nilai pointer Tumpuk.atas menjadi -1, yang mengindikasikan stack kosong
b. Menampilkan pesan konfirmasi "Tumpukan Kosong!"

Sama seperti fungsi hapus, fungsi ini tidak benar-benar menghapus data secara fisik dari array, melainkan hanya mengatur ulang nilai atas menjadi -1 sehingga stack dianggap kosong dan siap diisi kembali.

Table 9. Source Code Pembahasan Soal 2

```
void bersih() {
    Tumpuk.atas = -1;
    cout << "Tumpukan kosong!\n";
}
```

SOAL 3

A. Source Code

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>
#define n 10
using namespace std;

int PIL, F, R;
char PILIHAN[1], HURUF;
char Q[n];

void Inisialisasi() {
    F = 0;
    R = -1;
}

bool isFull() {
    return R == n - 1;
}

bool isEmpty() {
    return F > R;
}

void INSERT() {
    if (isFull()) {
        cout << "Queue Penuh!" << endl;
    } else {
        cout << "Masukkan 1 huruf: ";
        cin >> HURUF;
        R++;
        Q[R] = HURUF;
        cout << "Data '" << HURUF << "' berhasil dimasukkan ke dalam queue!" << endl;
    }
}

void DELETE() {
    if (isEmpty()) {
        cout << "Queue Kosong!" << endl;
    } else {
        cout << "Data '" << Q[F] << "' telah dihapus dari queue" << endl;
        F++;
        if (isEmpty()) {
            Inisialisasi();
            cout << "Queue telah direset" << endl;
        }
    }
}
```

Gambar 3. Source Code Analisis Soal 3

```
49  void CETAKLAYAR() {
50      if (isEmpty()) {
51          cout << "Queue Kosong!" << endl;
52      } else {
53          cout << "Isi Queue: ";
54          for (int i = F; i <= R; i++) {
55              cout << Q[i] << " ";
56          }
57          cout << endl;
58          cout << "Front: " << front << ", Rear: " << rear << endl;
59      }
60  }
61
62  int main() {
63      Inisialisasi();
64      do {
65          cout << "QUEUE" << endl;
66          cout << "===============" << endl;
67          cout << "1. INSERT" << endl;
68          cout << "2. DELETE" << endl;
69          cout << "3. CETAK QUEUE" << endl;
70          cout << "4. QUIT" << endl;
71          cout << "PILIHAN : ";
72          cin >> PILIHAN;
73          PIL = atoi(PILIHAN);
74
75          switch (PIL) {
76              case 1:
77                  INSERT();
78                  break;
79              case 2:
80                  DELETE();
81                  break;
82              case 3:
83                  CETAKLAYAR();
84                  break;
85              default:
86                  cout << "TERIMA KASIH" << endl;
87                  break;
88          }
89
90          cout << "Press any key to continue..." << endl;
91          getch();
92          system("cls");
93      } while (PIL < 4);
94      return 0;
95  }
```

Gambar 4. Source Code Analisis Soal 3

B. Pembahasan

1. Penambahan Library dan Namespace

   Bagian ini mendeklarasikan library yang dibutuhkan oleh program:

   a. <iostream>: Untuk operasi input/output standar seperti cin dan cout

b. <conio.h>: Menyediakan fungsi getch() untuk menahan tampilan layar
c. <stdlib.h>: Menyediakan fungsi system() untuk membersihkan layar
d. #define n 10: Mendefinisikan konstanta n dengan nilai 10 yang digunakan sebagai ukuran maksimum queue
e. using namespace std: Memungkinkan penggunaan fungsi dari namespace standard tanpa prefiks std::

Table 10. Source Code Pembahasan Soal 3

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>
#define n 10
using namespace std;
```

2. Deklarasi Variabel global

Bagian ini mendefinisikan konstanta dan variabel global yang digunakan dalam program:

a. PIL: Variabel integer untuk menyimpan pilihan menu setelah konversi
b. F (Front): Indeks untuk menunjukkan elemen terdepan dalam queue
c. R (Rear): Indeks untuk menunjukkan elemen terakhir dalam queue
d. PILIHAN[1]: Array karakter untuk menyimpan input pilihan menu (terbatas hanya 1 karakter)
e. HURUF: Variabel karakter untuk menyimpan data yang akan dimasukkan ke queue
f. Q[n]: Array dengan ukuran n untuk menyimpan elemen-elemen queue.

Table 11. Source Code Pembahasan Soal 3

```
int PIL, F, R;
char PILIHAN[1], HURUF;
char Q[n];
```

3. Fungsi Inisialisasi()

Fungsi ini menginisialisasi queue dengan menetapkan nilai awal untuk indeks front (F) dan rear (R):

a. Front (F) diatur ke 0, menunjukkan posisi awal untuk pengambilan data

b. Rear (R) diatur ke -1, menandakan queue kosong (belum ada elemen yang dimasukkan).

inisialisasi standar untuk queue linear, di mana indeks rear awalnya diatur lebih rendah dari front untuk menandakan kekosongan.

Table 12. Source Code Pembahasan Soal 3

```
void Inisialisasi() {
    F = 0;
    R = -1;
}
```

4. Fungsi isFull() dan isEmpty()

Dua fungsi boolean untuk memeriksa status queue:

a. isFull(): Mengembalikan true jika rear (R) telah mencapai indeks terakhir array (n-1)
b. isEmpty(): Mengembalikan true jika front (F) lebih besar dari rear (R).

Table 13. Source Code Pembahasan Soal 3

```
bool isFull() {
    return R == n - 1;
}

bool isEmpty() {
    return F > R;
}
```

5. Fungsi INSERT()

Fungsi INSERT() digunakan untuk memasukkan data ke dalam queue. Fungsi ini memiliki beberapa kondisi:

a. Jika stack penuh (isFull() == 1), maka ditampilkan pesan "Queue Penuh!”.
b. Jika kondisi lainnya (yaitu stack tidak penuh), "maka HURUF di input lalu nilai rear(R) ditambah 1 dan HURUF dimasukkan ke posisi rear(R), pesan konfirmasi akan ditampilkan dengan menunjukkan data yang dimasukkan.

Table 14. Source Code Pembahasan Soal 3

```
void INSERT() {
    if (isFull()) {
        cout << "Queue Penuh!" << endl;
    } else {
        cout << "Masukkan 1 huruf: ";
        cin >> HURUF;
        R++;
        Q[R] = HURUF;
        cout << "Data '" << HURUF << "' berhasil dimasukkan
ke dalam queue!" << endl;
    }
}
```

6. Fungsi DELETE()

Fungsi DELETE() bertugas untuk menghapus elemen teratas dari queue. Fungsi ini:

a. Jika tidak kosong, maka indeks pointer Tumpuk.atas dikurangi 1, yang secara efektif menghapus elemen teratas.
b. Jika kosong, maka ditampilkan pesan "Data Kosong".

Fungsi ini tidak benar-benar menghapus data secara fisik dari array, melainkan hanya mengubah indeks teratas yang dapat diakses.

Table 15. Source Code Pembahasan Soal 3

```
void DELETE() {
    if (isEmpty()) {
        cout << "Queue Kosong!" << endl;
    } else {
        cout << "Data '" << Q[F] << "' telah dihapus dari
queue" << endl;
        F++;
        if (isEmpty()) {
            Inisialisasi();
            cout << "Queue telah direset" << endl;
        }
    }
}
```

7. Fungsi CETAKLAYAR()

Fungsi CETAKLAYAR() bertugas untuk menampilkan seluruh isi queue. Fungsi ini:

a. Jika kosong, menampilkan pesan "Tumpukan Kosong".
b. Jika tidak kosong, menampilkan seluruh elemen queue mulai dari front hingga rear.

Penambahan informasi tentang nilai indeks front dan rear juga ditambahkan, sangat berguna untuk memahami status internal queue.

Table 16. Source Code Pembahasan Soal 3

```
void CETAKLAYAR() {
    if (isEmpty()) {
        cout << "Queue Kosong!" << endl;
    } else {
        cout << "Isi Queue: ";
        for (int i = F; i <= R; i++) {
            cout << Q[i] << " ";
        }
        cout << endl;
        cout << "Front: " << F << ", Rear: " << R << endl;
    }
}
```

8. Fungsi main()

Fungsi utama yang mengontrol alur program:

a. Memanggil Inisialisasi() untuk menyiapkan queue
b. Menampilkan menu interaktif dengan 4 pilihan
c. Membaca input pilihan pengguna dan mengkonversinya ke integer dengan atoi()
d. Menjalankan fungsi yang sesuai berdasarkan pilihan menggunakan struktur switch
e. Menunggu pengguna menekan tombol untuk melanjutkan
f. Membersihkan layar sebelum menampilkan menu kembali
g. Melanjutkan loop selama pilihan pengguna kurang dari 4

Table 17. Source Code Pembahasan Soal 3

```
int main() {
    Inisialisasi();
    do {
        cout << "QUEUE" << endl;
```

```
        cout << "===============" << endl;
        cout << "1. INSERT" << endl;
        cout << "2. DELETE" << endl;
        cout << "3. CETAK QUEUE" << endl;
        cout << "4. QUIT" << endl;
        cout << "PILIHAN : ";
        cin >> PILIHAN;
        PIL = atoi(PILIHAN);

        switch (PIL) {
            case 1:
                INSERT();
                break;
            case 2:
                DELETE();
                break;
            case 3:
                CETAKLAYAR();
                break;
            default:
                cout << "TERIMA KASIH" << endl;
                break;
        }

        cout << "Press any key to continue..." << endl;
        getch();
        system("cls");
    } while (PIL < 4);
    return 0;
}
```

GITHUB

https://github.com/IndraSuryadilaga/Algoritma-Dan-Struktur-Data