Telkom University Purwokerto

Aturan Praktikum Struktur Data

1. **Akun GitHub:** Setiap praktikan wajib memiliki akun GitHubyang aktif dan digunakan selama praktikum berlangsung.
2. **Invite Collaborator:** Setiap praktikan diwajibkan untuk menambahkan collaborator di setiap repository
    a. Asisten Praktikum: AndiniNH
    b. AsistenPraktikum:4ldiputra
3. **Repository Praktikum:** Setiap praktikan diwajibkan untuk membuat satu repository di GitHub yang akan digunakan untuk seluruh tugas dan laporan praktikum. Repository ini harus diatur dengan rapi dan sesuai dengan instruksi yang akan diberikan di lampiran.
4. **Penamaan Folder:** Penamaan folderdalam repositoryakan dibahas secara rinci di lampiran. Praktikan wajib mengikuti aturan penamaan yang telah ditentukan.

Nomor	Pertemuan	Penamaan
1	Penganalan Bahasa C++ Bagian Pertama	01_Pengenalan_CPP_Bagian_1
2	Pengenalan Bahasa C++ Bagian Kedua	02_Pengenalan_CPP_Bagian_2
3	Abstract Data Type	03_Abstract_Data_Type
4	Single Linked List Bagian Pertama	04_Single_Linked_List_Bagian_1
5	Single Linked List Bagian Kedua	05_Single_Linked_List_Bagian_2
6	Double Linked List Bagian Pertama	06_Double_Linked_List_Bagian_1
7	Stack	07_Stack
8	Queue	08_Queue
9	Assessment Bagian Pertama	09_Assessment_Bagian_1
10	Tree Bagian Pertama	10_Tree_Bagian_1
11	Tree Bagian Kedua	11_Tree_Bagian_2
12	Asistensi Tugas Besar	12_Asistensi_Tugas_Besar
13	Multi Linked List	13_Multi_Linked_List
14	Graph	14_Graph
15	Assessment Bagian Kedua	15_Assessment_Bagian_2
16	Tugas Besar	16_Tugas_Besar

5. **Jam Praktikum:**
    - Jam masuk praktikum adalah **1 jam lebih lambat** dari jadwal yang tercantum. Sebagai contoh, jika jadwalpraktikumadalah pukul 06.30 - 09.30, maka aturan praktikum akan diatur sebagai berikut:
        - **06.30 - 07.30**: Waktuini digunakan untuk **Tugas Praktikum dan Laporan Praktikum** yang dilakukan di luar laboratorium.
        - **07.30 - 0G.30**: Sesi ini mencakup **tutorial, diskusi, dan kasus problem- solving**. Kegiatan ini berlangsung di dalam laboratorium dengan alokasi waktu sebagai berikut:
            - **60 menit pertama**: Tugas terbimbing.
            - **60 menit kedua**: Tugas mandiri.
6. **Pengumpulan Tugasn Pendahuluan:** Tugas Pendahuluan (TP) wajib dikumpulkan melalui GitHub sesuai dengan format berikut:

    nama_repo/nama_pertemuan/TP_Pertemuan_Ke.md

    Sebagai contoh:

    STD_Yudha_Islalmi_Sulistya_XXXXXXXX/01_Running_Modul/TP_01.md

7. **Pengecekan Tugas Pendahuluan:** Pengumpulan laporan praktikum akan diperiksa **1 hari sebelum praktikum selanjutnya** dimulai. Pastikan tugas telah diunggah tepat waktu untuk menghindari sanksi.

LAPORAN PRAKTIKUM
MODUL 7
STACK

Disusun Oleh :

Zaenarif Putra ‘Ainurdin – 2311104049

Kelas :

SE-07-02

Dosen :

Wahyu Andi Saputra, S.pd,M.Eng

PROGRAM STUDI SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

I. TUJUAN

1. Mampu memahami konsep stack pada struktur data dan algoritma
2. Mampu mengimplementasikan operasi-operasi pada stack
3. Mampu memecahkan permasalahan dengan solusi stack

II. LANDASAN TEORI

Tumpukan, atau tumpukan, adalah struktur data yang menggunakan prinsip Last-In, First-Out (LIFO), di mana elemen yang dimasukkan terakhir akan diambil pertama. Push (menambahkan elemen), pop (menghapus elemen teratas), dan top adalah operasi utama pada stack. Operasi tambahan seperti isEmpty, isFull, size, peek, clear, dan search membantu fungsi tambahan, seperti memeriksa kondisi stack, mengelola elemen, dan melakukan pencarian. Aplikasi seperti manajemen memori, evaluasi ekspresi aritmatika, dan manajemen panggilan fungsi menggunakan Stack, yang menjadikannya salah satu struktur data yang efektif dan penting dalam pengembangan perangkat lunak.

III. GUIDE

1. Guide_1

   a. Syntax

```
#include <iostream>
#define MAX 100

using namespace std;

class Stack {
   int top;
   int arr[MAX];

public:
   Stack() { top = -1; }
   bool isFull() { return top == MAX - 1; }
   bool isEmpty() { return top == -1; }

   void push(int x) {
      if (isFull()) {
         cout << "Stack Overflow\n";
         return;
      }
      arr[++top] = x;
   }

   void pop() {
      if (isEmpty()) {
```

```
            cout << "Stack Underflow\n";
            return;
        }
        top--;
    }

    int peek() {
        if (!isEmpty()) {
            return arr[top];
        }
        cout << "Stack is empty\n";
        return -1;
    }

    void display() {
        if (isEmpty()) {
            cout << "Stack is empty\n";
            return;
        }
        for (int i = top; i >= 0; i--) {
            cout << arr[i] << " ";
        }
        cout << "\n";
    }
};

int main() {
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);

    cout << "Stack elements : ";
    s.display();

    cout << "Top element : " << s.peek() << "\n";

    s.pop();
    s.pop();
    cout << "After popping, stack elements : ";
    s.display();

    return 0;
}
```

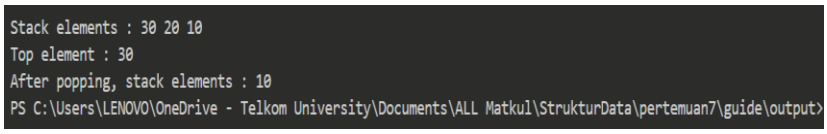
b. Penjelasan syntax

- Class Guide :

  - #include <iostream>: Ini adalah perintah preprocessor yang menghubungkan ke file header iostream, yang berisi definisi untuk input dan output dasar.
  - #define MAX 100: Ini adalah perintah preprocessor yang mendefinisikan simbol MAX dengan nilai 100.
  - using namespace std;: Ini mengaktifkan namespace std, sehingga Anda tidak perlu mengetik std:: pada setiap kata kunci C++ yang berasal dari namespace tersebut.
  - class Stack { ... };: Ini adalah definisi dari sebuah class bernama Stack. Class ini akan memuat atribut dan method yang berkaitan dengan operasi stack.
  - int top;: Ini adalah atribut dari class Stack, yang digunakan sebagai indeks untuk elemen teratas pada stack.
  - int arr[MAX];: Ini adalah atribut dari class Stack, yang merupakan sebuah array yang akan digunakan untuk menyimpan elemen-elemen pada stack.
  - bool isFull() { ... }: Ini adalah method dari class Stack, yang digunakan untuk menentukan apakah stack penuh atau belum.
  - bool isEmpty() { ... }: Ini adalah method dari class Stack, yang digunakan untuk menentukan apakah stack kosong atau tidak.
  - void push(int x) { ... }: Ini adalah method dari class Stack, yang digunakan untuk menambahkan sebuah elemen ke dalam stack.
  - void pop() { ... }: Ini adalah method dari class Stack, yang digunakan untuk menghapus elemen teratas pada stack.
  - int peek() { ... }: Ini adalah method dari class Stack, yang digunakan untuk mendapatkan nilai dari elemen teratas pada stack tanpa menghapusnya.
  - void display() { ... }: Ini adalah method dari class Stack, yang digunakan untuk mencetak semua elemen pada stack.
  - Stack s;: Ini adalah deklarasi dan pembuatan sebuah objek bernama s dari class Stack.
  - s.push(10);: Ini adalah pemanggilan method push pada objek s, dengan argumen 10.
  - s.push(20);: Ini adalah pemanggilan method push pada objek s, dengan argumen 20.
  - s.push(30);: Ini adalah pemanggilan method push pada objek s, dengan argumen 30.
  - cout << "Stack elements : ";: Ini adalah perintah untuk mencetak teks "Stack elements : " ke layar.
  - s.display();: Ini adalah pemanggilan method display pada objek s.
  - cout << "Top element : " << s.peek() << "\n";: Ini adalah perintah untuk mencetak teks "Top element : " ke layar, lalu mencetak nilai dari elemen teratas pada stack, dan akhirnya mencetak simbol baris baru "\n".
  - s.pop();: Ini adalah pemanggilan method pop pada objek s.

- s.pop();: Ini adalah pemanggilan method pop pada objek s.
- cout << "After popping, stack elements : ";: Ini adal ah perintah untuk mencetak teks "After popping, stack elements : " ke layar.
- s.display();: Ini adalah pemanggilan method display pada objek s untuk menampilkan elemen-elemen yang tersisa di stack setelah dua elemen dihapus.
- return 0;: Ini adalah pernyataan yang menandakan bahwa program telah selesai dijalankan dengan sukses. 1. #include <iostream>: Ini adalah perintah preprocessor yang menghubungkan ke file header iostream, yang berisi definisi untuk input dan output dasar.

c. Output



2. Guide_2

a. Syntax :

```
#include <iostream>

using namespace std;

class Node {
public:
    int data;
    Node* next;

    Node(int value) {
        data = value;
        next = nullptr;
    }
};

class Stack {
private:
    Node* top;

public:
    Stack() { top = nullptr; }

    bool isEmpty() { return top == nullptr; }

    void push(int x) {
        Node* newNode = new Node(x);
        newNode->next = top;
        top = newNode;
```

```
    }

    void pop() {
        if (isEmpty()) {
            cout << "Stack Underflow\n";
            return;
        }
        Node* temp = top;
        top = top->next;
        delete temp;
    }

    int peek() {
        if (!isEmpty()) {
            return top->data;
        }
        cout << "Stack is empty\n";
        return -1;
    }

    void display() {
        if (isEmpty()) {
            cout << "Stack is empty\n";
            return;
        }
        Node* current = top;
        while (current) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }
};

int main() {
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);

    cout << "Stack elements: ";
    s.display();

    cout << "Top element: " << s.peek() << "\n";

    s.pop();
    s.pop();

    cout << "After popping, stack elements: ";
    s.display();
```

```
    return 0;
}
```

b. Penjelasan Syntax :

- Class Guide2 :

    - class Node { ... };: Ini adalah definisi dari sebuah class bernama Node.
    - public:: Ini adalah access modifier yang menentukan bahwa atribut dan method yang diikuti dapat diakses dari luar class.
    - int data;: Ini adalah atribut dari class Node, yang digunakan untuk menyimpan nilai integer.
    - Node* next;: Ini adalah atribut dari class Node, yang digunakan untuk menyimpan pointer ke Node lainnya.
    - Node(int value) { ... }: Ini adalah constructor dari class Node, yang digunakan untuk membuat objek Node baru dengan nilai yang diberikan.
    - class Stack { ... };: Ini adalah definisi dari sebuah class bernama Stack.
    - private:: Ini adalah access modifier yang menentukan bahwa atribut dan method yang diikuti hanya dapat diakses dari dalam class.
    - Node* top;: Ini adalah atribut dari class Stack, yang digunakan untuk menyimpan pointer ke Node teratas pada stack.
    - Stack() { top = nullptr; }: Ini adalah constructor dari class Stack, yang digunakan untuk membuat objek Stack baru dengan top pointer yang diinisialisasi ke nullptr.
    - bool isEmpty() { ... }: Ini adalah method dari class Stack, yang digunakan untuk menentukan apakah stack kosong atau tidak.
    - void push(int x) { ... }: Ini adalah method dari class Stack, yang digunakan untuk menambahkan sebuah elemen ke dalam stack.
    - void pop() { ... }: Ini adalah method dari class Stack, yang digunakan untuk menghapus elemen teratas pada stack.
    - int peek() { ... }: Ini adalah method dari class Stack, yang digunakan untuk mendapatkan nilai dari elemen teratas pada stack tanpa menghapusnya.
    - void display() { ... }: Ini adalah method dari class Stack, yang digunakan untuk mencetak semua elemen pada stack.
    - void push(int x) { ... }: Ini adalah method push dari class Stack.
    - Node* newNode = new Node(x);: Ini membuat objek Node baru dengan nilai x dan menyimpannya dalam pointer newNode.
    - newNode->next = top;: Ini mengatur next pointer dari newNode ke top pointer, sehingga newNode menjadi elemen teratas pada stack.
    - top = newNode;: Ini mengatur top pointer ke newNode, sehingga newNode menjadi elemen teratas pada stack.
    - void pop() { ... }: Ini adalah method pop dari class Stack.
    - if (isEmpty()) { ... }: Ini memeriksa apakah stack kosong atau

tidak. Jika kosong, maka akan mencetak pesan error.

- Node* temp = top;: Ini membuat pointer temp yang menunjuk ke elemen teratas pada stack.
- top = top->next;: Ini mengatur top pointer ke elemen berikutnya pada stack.
- delete temp;: Ini menghapus elemen teratas pada stack dengan menggunakan pointer temp.
- int peek() { ... }: Ini adalah method peek dari class Stack.
- if (!isEmpty()) { ... }: Ini memeriksa apakah stack kosong atau tidak. Jika tidak kosong, maka akan mengembalikan nilai dari elemen teratas pada stack.
- return top->data;: Ini mengembalikan nilai dari elemen teratas pada stack.
- cout << "Stack is empty\n";: Ini mencetak pesan error jika stack kosong.
- return -1;: Ini mengembalikan nilai -1 jika stack kosong.
- void display() { ... }: Ini adalah method display dari class Stack.
- if (isEmpty()) { ... }: Ini memeriksa apakah stack kosong atau tidak. Jika kosong, maka akan mencetak pesan error.
- Node* current = top;: Ini membuat pointer current yang menunjuk ke elemen teratas pada stack.
- while (current) { ... }: Ini adalah loop yang akan mencetak semua elemen pada stack.
- cout << current->data << " ";: Ini mencetak nilai dari elemen pada stack.
- current = current->next;: Ini mengatur pointer current ke elemen berikutnya pada stack.
- cout << endl;: Ini mencetak simbol baris baru.
- Stack s;: Ini membuat objek Stack baru bernama s.
- s.push(10);: Ini memanggil method push pada objek s untuk menambahkan elemen 10 ke dalam stack.
- s.push(20);: Ini memanggil method push pada objek s untuk menambahkan elemen 20 ke dalam stack.
- s.push(30);: Ini memanggil method push pada objek s untuk menambahkan elemen 30 ke dalam stack.
- cout << "Stack elements: ";: Ini mencetak teks "Stack elements: " ke layar.
- s.display();: Ini memanggil method display pada objek s untuk menampilkan semua elemen yang ada di dalam stack.
- cout << "Top element: " << s.peek() << "\n";: Ini mencetak teks "Top element: " diikuti dengan nilai dari elemen teratas pada stack.
- s.pop();: Ini memanggil method pop pada objek s untuk menghapus elemen teratas dari stack.
- s.pop();: Ini memanggil method pop pada objek s untuk menghapus elemen teratas dari stack lagi.
- cout << "After popping, stack elements: ";: Ini mencetak teks "After popping, stack elements: " ke layar.
- s.display();: Ini memanggil method display pada objek s untuk menampilkan elemen-elemen yang tersisa di stack setelah dua

elemen dihapus.

- return 0;: Ini menandakan bahwa program telah selesai dijalankan dengan sukses.

c. Output :

```
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan7\guide\output>
```

IV. UNGUIDED

1. Task1

a. Syntax :

```
#include <iostream>
#include <stack>
#include <string>

using namespace std;

bool isPalindrome(const string &str) {
    stack<char> s;
    string cleanedstr;

    for (char ch : str) {
        if (isalnum(ch)) {
            cleanedstr += tolower(ch);
        }
    }

    for (char ch : cleanedstr) {
        s.push(ch);
    }

    for (char ch : cleanedstr) {
        if (s.top() != ch) {
            return false;
        }
        s.pop();
    }

    return true;
}

int main() {
    string input;
    cout << "Masukkan Sebuah Kalimat : ";
```

```
    getline(cin, input);

    if (isPalindrome(input)) {
        cout << "Kalimat adalah Palindrome.\n";
    }
    else {
        cout << "Bukan Palindrome.\n";
    }

    return 0;
}
```

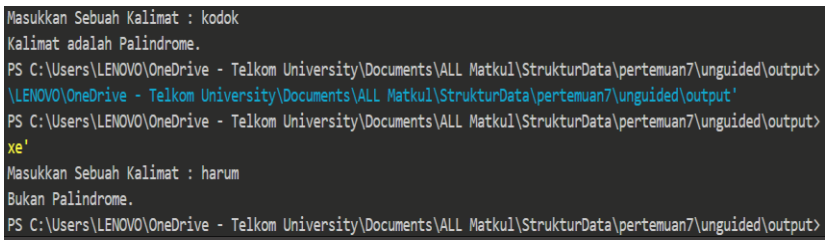
b. Penjelasan Syntax

- bool isPalindrome(const string &str) { ... }: Ini adalah definisi fungsi isPalindrome yang menerima parameter str bertipe string dan mengembalikan nilai boolean (true atau false). Fungsi ini digunakan untuk memeriksa apakah string yang diberikan adalah palindrome.
- stack<char> s;: Ini mendeklarasikan sebuah stack bertipe char yang akan digunakan untuk menyimpan karakter dari string yang telah dibersihkan.
- string cleanedstr;: Ini mendeklarasikan sebuah string kosong yang akan digunakan untuk menyimpan karakter-karakter dari str yang telah dibersihkan (hanya karakter alfanumerik dan dalam huruf kecil).
- for (char ch : str) { ... }: Ini adalah loop yang akan iterasi setiap karakter dalam string str.
- if (isalnum(ch)) { ... }: Ini memeriksa apakah karakter ch adalah alfanumerik (angka atau huruf). Jika ya, maka karakter tersebut akan diproses lebih lanjut.
- cleanedstr += tolower(ch);: Ini menambahkan karakter ch yang telah diubah menjadi huruf kecil ke dalam string cleanedstr.
- for (char ch : cleanedstr) { s.push(ch); }: Ini adalah loop yang akan iterasi setiap karakter dalam cleanedstr dan memasukkan setiap karakter ke dalam stack s.
- for (char ch : cleanedstr) { ... }: Ini adalah loop kedua yang akan iterasi setiap karakter dalam cleanedstr untuk memeriksa apakah string tersebut adalah palindrome.
- if (s.top() != ch) { return false; }: Ini memeriksa apakah karakter teratas pada stack s sama dengan karakter saat ini dari cleanedstr. Jika tidak sama, maka fungsi akan mengembalikan false, yang berarti string bukan palindrome.
- s.pop();: Ini menghapus karakter teratas dari stack s setelah perbandingan.
- return true;: Jika semua karakter cocok, maka fungsi mengembalikan true, yang berarti string adalah palindrome.
- int main() { ... }: Ini adalah fungsi utama yang menjadi titik awal eksekusi program.
- string input;: Ini mendeklarasikan sebuah variabel string bernama input untuk menyimpan kalimat yang dimasukkan oleh pengguna.
- cout << "Masukkan Sebuah Kalimat : ";: Ini mencetak teks "Masukkan

Sebuah Kalimat : " ke layar sebagai prompt untuk pengguna.

- getline(cin, input);: Ini membaca seluruh baris input dari pengguna dan menyimpannya dalam variabel input.
- if (isPalindrome(input)) { ... }: Ini memanggil fungsi isPalindrome dengan argumen input. Jika hasilnya true, maka blok kode di dalamnya akan dieksekusi.
- cout << "Kalimat adalah Palindrome.\n";: Jika string adalah palindrome, maka akan mencetak pesan bahwa kalimat tersebut adalah palindrome.
- else { cout << "Bukan Palindrome.\n"; }: Jika string bukan palindrome, maka akan mencetak pesan bahwa kalimat tersebut bukan palindrome.
- return 0;: Ini menandakan bahwa program telah selesai dijalankan dengan sukses.

c. Output



2. Task2

a. Syntax

```
#include <iostream>
#include <string>
#include <stack>
#include <sstream>

using namespace std;

void reverseWords(const string &sentence) {
   stringstream ss(sentence);
   string word;

   cout << "Datastack Array :\nData : ";
   while (ss >> word) {
      stack<char> s;

      for (char ch : word) {
         s.push(ch);
      }

      while (!s.empty()) {
         cout << s.top();
         s.pop();
      }
```

```
            cout << " ";
        }
        cout << endl;
    }

    int main() {
        string input;
        cout << "Masukkan Sebuah Kalimat Dengan Minimal 3 Kata :
    ";
        getline(cin, input);

        reverseWords(input);

        return 0;
    }
```

b. Penjelasan Syntax

- void reverseWords(const string &sentence) { ... }: Ini adalah definisi fungsi reverseWords yang menerima parameter sentence bertipe string dan tidak mengembalikan nilai apa-apa. Fungsi ini digunakan untuk membalikkan kata-kata dalam sebuah kalimat.
- stringstream ss(sentence);: Ini membuat sebuah stringstream bernama ss yang berisi kalimat yang diberikan.
- string word;: Ini mendeklarasikan sebuah string kosong bernama word yang akan digunakan untuk menyimpan setiap kata dalam kalimat.
- cout << "Datastack Array :\nData : ";: Ini mencetak teks "Datastack Array :\nData : " ke layar sebagai header untuk output.
- while (ss >> word) { ... }: Ini adalah loop yang akan iterasi setiap kata dalam kalimat yang diberikan.
- stack<char> s;: Ini mendeklarasikan sebuah stack bertipe char yang akan digunakan untuk menyimpan karakter-karakter dalam setiap kata.
- for (char ch : word) { s.push(ch); }: Ini adalah loop yang akan iterasi setiap karakter dalam kata saat ini dan memasukkan setiap karakter ke dalam stack s.
- while (!s.empty()) { ... }: Ini adalah loop yang akan iterasi setiap karakter dalam stack s dan mencetaknya dalam urutan terbalik.
- cout << s.top();: Ini mencetak karakter teratas dalam stack s.
- s.pop();: Ini menghapus karakter teratas dari stack s.
- cout << " ";: Ini mencetak spasi untuk memisahkan setiap kata.
- int main() { ... }: Ini adalah fungsi utama yang menjadi titik awal eksekusi program.
- string input;: Ini mendeklarasikan sebuah variabel string bernama input untuk menyimpan kalimat yang dimasukkan oleh pengguna.
- cout << "Masukkan Sebuah Kalimat Dengan Minimal 3 Kata : ";: Ini mencetak teks "Masukkan Sebuah Kalimat Dengan Minimal 3 Kata : " ke layar sebagai prompt untuk pengguna.
- getline(cin, input);: Ini membaca seluruh baris input dari pengguna dan menyimpannya dalam variabel input.
- reverseWords(input);: Ini memanggil fungsi reverseWords dengan

argumen input untuk membalikkan kata-kata dalam kalimat.

- return 0;: Ini menandakan bahwa program telah selesai dijalankan dengan sukses.

c. Output

```
Masukkan Sebuah Kalimat Dengan Minimal 3 Kata : Telkom Purwokerto
Datastack Array :
Data : mokleT otrekowruP
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan7\unguided\output>
```

V. KESIMPULAN

Stack adalah struktur data yang menerapkan prinsip Last-In, First-Out (LIFO), di mana elemen terakhir yang dimasukkan adalah yang pertama diambil. Operasi dasar pada stack meliputi push (menambahkan elemen), pop (menghapus elemen), peek (melihat elemen teratas), serta pemeriksaan apakah stack kosong atau penuh. Stack memiliki berbagai aplikasi, seperti manajemen memori, evaluasi ekspresi aritmatika, dan manajemen panggilan fungsi dalam pemrograman.