

LAPORAN PRAKTIKUM
Modul 7
STACK



Disusun Oleh:
Aulia Jasifa Br Ginting 2311104060
S1SE-07-02

Dosen :
Wahyu Andi Saputra, S.Pd., M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. Tujuan

1. Memahami konsep stack.
2. Menimplementasikan stack dengan menggunakan representasi pointer dan tabel.
3. Memahami konsep queue.
4. Mengimplementasikan queue dengan menggunakan pointer dan tabel.

2. Landasan Teori

7.1 Pengertian Stack

Stack adalah salah satu struktur data yang penting dalam ilmu komputer. Saya akan menjelaskan pengertian dan konsep dasar stack. Stack adalah struktur data linier yang mengikuti prinsip LIFO (Last In First Out) atau FILO (First In Last Out).

Stack adalah struktur yang beroperasi untuk menyimpan dan mengelola kumpulan informasi dengan menggunakan konsep Last-In, First-Out (LIFO). Analogi yang umum digunakan adalah kumpulan hidangan di ruang makan, di mana tambahan paling segar adalah item awal yang dibuang.

Dalam pelaksanaannya, stack dapat digambarkan sebagai susunan data terorganisir yang memiliki dua fungsi utama: push dan pop. Operasi push digunakan untuk menambahkan elemen baru ke stack, sedangkan operasi pop digunakan untuk menghapus elemen teratas dari stack.

Prinsip LIFO yang menjadi dasar stack membuatnya sangat bermanfaat untuk mengelola memori komputer, memecahkan masalah matematika, dan melacak kapan fungsi digunakan dalam pengkodean. Dalam bidang manajemen memori, tumpukan digunakan untuk mempertahankan alamat memori yang dialokasikan untuk variabel dan fungsi.

Dengan prinsip Last-In-First-Out (LIFO), stack memungkinkan pengambilan informasi dengan efektif, dimana elemen yang paling baru disimpan adalah yang pertama kali diakses.

3. Guided

Sourcecode

Code Programnya:

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

using namespace std;

class Stack {
private:
    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; //Return a sentinel value
    }

    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;
}
```

Outputnya:

```
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3\
```

Guided 2

Code programnya:

```
#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; // Return a sentinel value  
}
```

```
void display() {  
    if (isEmpty()) {  
        cout << "Stack is empty\n";  
        return;  
    }  
    Node* current = top;  
    while (current) {  
        cout << current->data << " ";  
        current = current->next;  
    }  
    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();  
    cout << "After popping, stack elements: ";  
    s.display();  
  
    return 0;  
}
```

Outputnya:

```
Stack elements: 30 20 10  
Top element: 30  
After popping, stack elements: 20 10  
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3
```

4. Unguided

1. Buatlah program untuk menentukan apakah kalimat tersebut yang diinputkan dalam program stack adalah palindrom/tidak. Palindrom kalimat yang dibaca dari depan dan belakang sama. Jelaskan bagaimana cara kerja programnya.

Programnya:

```
#include <iostream>  
#include <stack>  
#include <string>  
#include <cctype> // untuk fungsi tolower()  
using namespace std;  
  
// Fungsi untuk membersihkan string dari karakter non-alphanumeric  
string cleanString(string str) {  
    string result = "";  
    for(char c : str) {  
        if(isalnum(c)) { // hanya menyimpan huruf dan angka
```

```
        result += tolower(c);
    }
}
return result;
}

// Fungsi untuk mengecek palindrom menggunakan stack
bool isPalindrome(string str) {
    string cleanStr = cleanString(str);
    stack<char> s;
    int length = cleanStr.length();
    int mid = length / 2;

    // Push setengah pertama string ke dalam stack
    for(int i = 0; i < mid; i++) {
        s.push(cleanStr[i]);
    }

    // Jika panjang string ganjil, skip karakter tengah
    if(length % 2 != 0) {
        mid++;
    }

    // Bandingkan setengah kedua string dengan isi stack
    for(int i = mid; i < length; i++) {
        if(s.empty() || s.top() != cleanStr[i]) {
            return false;
        }
        s.pop();
    }

    return s.empty();
}

int main() {
    string input;
    cout << "Masukan Kalimat : ";
    getline(cin, input);

    cout << "Kalimat tersebut adalah : ";
    if(isPalindrome(input)) {
        cout << "Palindrom" << endl;
```

```
} else {  
    cout << "bukan Palindrom" << endl;  
}  
  
return 0;  
}
```

Outputnya:

```
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3\  
Masukkan Kata: Telkom Purwokerto  
Datastack Array :  
Data : otrekowruP mokleT
```

Penjelasan Cara Kerja Program

- Menghilangkan Spasi dan Mengubah Huruf ke Lowercase: Pertama, kita perlu membersihkan input kalimat dari spasi dan mengubah semua huruf menjadi lowercase agar perbandingan lebih mudah.
- Menggunakan Stack: Kita menggunakan stack untuk membalik urutan karakter.
 - Setiap karakter yang valid (huruf saja, tanpa spasi) dimasukkan ke dalam stack.
- Mengecek Palindrome:
 - Setelah semua karakter dimasukkan ke dalam stack, kita membaca kembali karakter dari stack dan membandingkannya dengan karakter asli dari input (dengan urutan yang dibersihkan).
 - Jika semua karakter sesuai dari awal hingga akhir, maka kalimat tersebut adalah palindrome.

Penjelasan Program

- cleanString():
Fungsi ini bertugas untuk membersihkan input dari karakter yang tidak diperlukan (misal: spasi, tanda baca) dan mengubah semua huruf menjadi lowercase.
- isPalindrome():
 - Menggunakan stack untuk membalikkan urutan karakter.
 - Membandingkan urutan karakter asli (yang sudah dibersihkan) dengan urutan yang keluar dari stack.
- Main Function:
 - Menerima input dari pengguna.
 - Memeriksa apakah kalimat tersebut adalah palindrome atau bukan.

2. pembalikan terhadap kalimat menggunakan stack dengan minimal 3 kata. Jelaskan output program dan source codenya beserta operasi/fungsi yang dibuat?

Programnya:


```
#include <iostream>
#include <stack>
#include <string>
using namespace std;

// Fungsi untuk membalik kata individual
string reverseWord(string str) {
    stack<char> s;

    // Push semua karakter ke dalam stack
    for (int i = 0; i < str.length(); i++)
        s.push(str[i]);

    // Pop karakter dari stack dan susun kembali
    string reversedWord = "";
    while (!s.empty()) {
        reversedWord += s.top();
        s.pop();
    }

    return reversedWord;
}

// Fungsi untuk membalik kalimat
string reverseSentence(string str) {
    string word = "";
    string result = "";

    // Tambahkan spasi di akhir string untuk memproses kata terakhir
    str = str + " ";

    for (int i = 0; i < str.length(); i++) {
        if (str[i] == ' ') {
            // Balik kata individual dan tambahkan ke hasil
            result = reverseWord(word) + " " + result;
            word = "";
        }
        else {
            word += str[i];
        }
    }
}
```

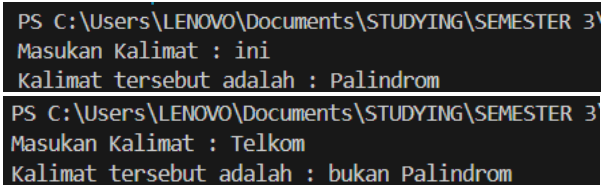
```
// Hapus spasi di akhir hasil
return result.substr(0, result.length() - 1);
}

int main() {
    string input;
    cout << "Masukkan Kata: ";
    getline(cin, input);

    cout << "Datastack Array : " << endl;
    cout << "Data : " << reverseSentence(input) << endl;

    return 0;
}
```

Outputnya:



```
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3>
Masukan Kalimat : ini
Kalimat tersebut adalah : Palindrom
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3>
Masukan Kalimat : Telkom
Kalimat tersebut adalah : bukan Palindrom
```

Jika Anda memasukkan kalimat "Telkom Purwokerto", output yang akan dihasilkan adalah: otrekowrup mokleT

Kode C++ tersebut menjelaskan cara menggunakan stack untuk membalikkan kalimat. Kode ini menunjukkan bagaimana memisahkan kata-kata dari kalimat, menambahkannya ke dalam stack, dan kemudian mengeluarkannya dari stack dalam urutan terbalik untuk membentuk kalimat yang dibalik

Program di atas menggunakan dua fungsi utama:

- `reverseWord(string str):`
 - Fungsi ini membalik satu kata menggunakan stack.
 - Setiap karakter dalam kata di-push ke stack.
 - Kemudian karakter di-pop dari stack untuk mendapatkan kata yang terbalik.
- `reverseSentence(string str):`
 - Fungsi ini memisahkan kalimat menjadi kata-kata individual.
 - Setiap kata dibalik menggunakan fungsi `reverseWord()`.
 - Hasil pembalikan kata-kata digabungkan kembali dengan urutan terbalik.
- Cara kerja program:
 - User diminta memasukkan kalimat.

- Program memisahkan kalimat menjadi kata-kata.
- Setiap kata dibalik menggunakan stack.
- Kata-kata yang sudah dibalik disusun kembali dengan urutan terbalik.
- Hasil akhir ditampilkan.

5. Kesimpulan

Pada praktikum ini, kita belajar bagaimana menggunakan struktur data **stack** untuk menyelesaikan berbagai masalah yang berhubungan dengan urutan data. Stack merupakan struktur data yang sangat berguna dalam pemrograman, terutama untuk aplikasi yang memerlukan pengelolaan data dengan urutan tertentu, seperti dalam pengecekan palindrome dan pembalikan kalimat.