LAPORAN PRAKTIKUM
Modul 7
“Stack”

Disusun Oleh:
Dimastian Aji Wibowo (2311104058)
SE-07-02

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

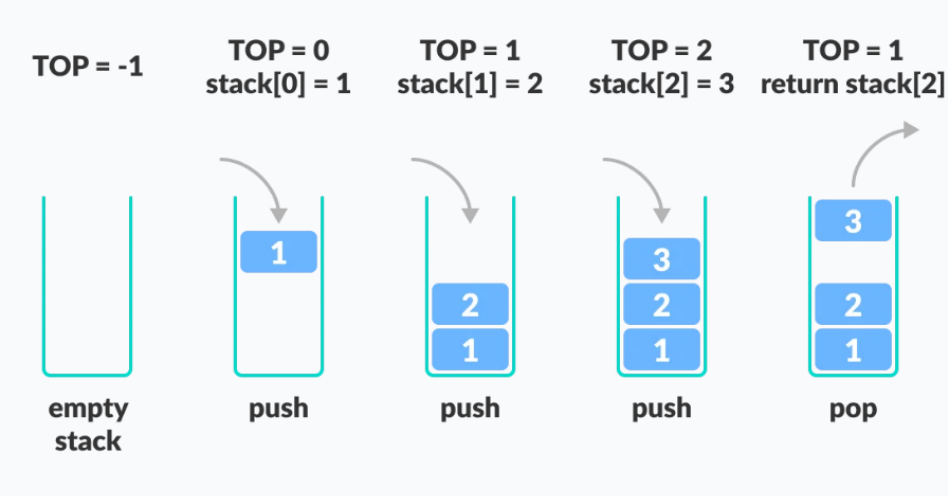
1. **Tujuan**
    - Memahami konsep stack pada struktur data dan algoritma.
    - Mampu mengimplementasikan stack dengan menggunakan representasi pointer dan tabel.
    - Mampu mengimplementasikan operasi-operasi pada stack
    - Mampu memecahkan permasalahan dengan solusi stack
2. **Landasan Teori**

    Sebuah stack atau tumpukan merupakan struktur data yang berfungsi untuk menyimpan dan mengelola kumpulan data dengan prinsip Last-In, First-Out (LIFO). Analogi yang sering digunakan adalah tumpukan piring di kafetaria, di mana piring terakhir yang ditambahkan akan menjadi yang pertama diambil.

    Dalam implementasinya, stack dapat direpresentasikan sebagai struktur data terurut yang memiliki dua operasi utama: push dan pop. Operasi push digunakan untuk menambahkan elemen baru ke dalam stack, sementara operasi pop digunakan untuk menghapus elemen teratas dari stack.

    Prinsip LIFO yang menjadi dasar stack membuatnya sangat bermanfaat dalam berbagai aplikasi, termasuk manajemen memori komputer, evaluasi ekspresi aritmatika, dan manajemen panggilan fungsi dalam pemrograman. Sebagai contoh, dalam manajemen memori, stack digunakan untuk menyimpan alamat-alamat memori yang dialokasikan untuk variabel dan fungsi.

    Dengan prinsip LIFO ini, stack memungkinkan akses data dengan efisiensi, di mana elemen yang terakhir dimasukkan akan menjadi yang pertama diambil. Hal ini menjadikannya salah satu struktur data yang sangat penting dalam pengembangan perangkat lunak dan pemrograman komputer.



Working of Stack Data Structure

Operasi pada stack melibatkan beberapa fungsi dasar yang dapat dilakukan pada struktur data ini. Berikut adalah beberapa operasi umum pada stack:

a. Push (Masukkan): Menambahkan elemen ke dalam tumpukan pada posisi paling atas atau ujung.
b. Pop (Keluarkan): Menghapus elemen dari posisi paling atas atau ujung tumpukan.
c. Top (Atas): Mendapatkan nilai atau melihat elemen teratas pada tumpukan tanpa menghapusnya.
d. IsEmpty (Kosong): Memeriksa apakah tumpukan kosong atau tidak.
e. IsFull (Penuh): Memeriksa apakah tumpukan penuh atau tidak (terutama pada implementasi tumpukan dengan kapasitas terbatas).
f. Size (Ukuran): Mengembalikan jumlah elemen yang ada dalam tumpukan.
g. Peek (Lihat): Melihat nilai atau elemen pada posisi tertentu dalam tumpukan tanpa menghapusnya.
h. Clear (Hapus Semua): Mengosongkan atau menghapus semua elemen dari tumpukan.
i. Search (Cari): Mencari keberadaan elemen tertentu dalam tumpukan.

3. Guided

A. Guided 1

1. Mendefinisikan MAX dengan nilai 100 untuk menentukan ukuran maksimum stack, mendefinisikan class Stack dengan atribut int top

untuk menyimpan indeks elemen teratas pada stack dengan nilai awal -1 yang menunjukan stack kosong, dan atribut arr[MAX] yaitu ukuran dari array untuk menyimpan elemen – elemen stack.

2. Membuat konstruktor class Stack dan menginisialisasi top ke -1 yang menandakan stack kosong.
3. Membuat method isFull() dengan menggunakan boolean untuk memeriksa apakah stack sudah penuh dengan cara mengembalikan nilai true jika top sudah mencapai MAX-1.
4. Membuat method isEmpty() dengan menggunakan boolean untuk memeriksa apakah stack kosong dengan cara mengembalikan nilai true jika top bernilai -1.
5. Method push() untuk menambahkan elemen kedalam stack dengan memeriksa terlebih dahulu apakah stack penuh atau tidak dengan menggunakan method isFull() dan jika sudah penuh maka tampilkan pesan bahwa stack penuh, jika stack belum penuh maka elemen ditambahkan ke top+1.
6. Method pop() untuk menghapus elemen pada top dengan memeriksa apakah stack kosong menggunakan method isEmpty() dan jika stack kosong maka tampilkan pesan bahwa stack kosong, jika stack tidak kosong maka top-1.
7. Method peek() untuk mengembalikan nilai top tanpa menghapusnya dengan memeriksa apakah stack kosong menggunakan isEmpty() dan jika stack kosong tampilkan pesan stack kosong lalu mengembalikan -1 dan jika stack tidak kosong maka mengembalikan arr[top].

```
#define MAX 100
#include <iostream>
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";
        }
        top --;
    }
    int peek(){
        if(isEmpty()){
            cout<<"Stack is empty\n";
            return -1;
        }
        return arr[top];
    }
```

8. Method display() untuk menampilkan semua elemen dalam stack dari awal sampai terakhir dengan memeriksa apakah stack kosong menggunakan method isEmpty() dan jika kosong maka tampilkan pesan stack kosong dan jika tidak kosong maka menampilkan elemen dalam stack menggunakan perulangan.
9. Membuat main() dengan menginisialisasi class Stack menjadi s, menambahkan elemen kedalam stack dengan push(), menghapus elemen dari stack menggunakan pop(), dan menampilkan isi stack menggunakan display().

```
    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. Guided 2

1. Mendefinisikan struktur node dengan atribut int data untuk menyimpan nilai integer untuk setiap node Node* next yang merupakan pointer yang menunjuk ke node berikutnya.
2. Membuat konstruktor untuk menginisialisasi data dengan nilai value dan mengatur next ke nullptr.
3. Mendefinisikan class Stack dengan top merupakan pointer ke node teratas, membuat variabel Node* top yang merupakan pointer ke node teratas dalam stack untuk akses ke elemen paling atas, membuat konstruktor class Stack dengan menginisialisasi stack kosong dengan top bernilai nullptr.
4. Method isEmpty() menggunakan boolean untuk memeriksa apakah stack kosong dengan mengembalikan true dengan cara memeriksa apakah top adalah nullptr atau bukan.
5. Method push(int x) untuk menambahkan elemen x kedalam stack dengan membuat node baru newNode dengan nilai x, mengatur newNode->next ke top, dan mengubah newNode menjadi top.
6. Method pop() untuk menghapus elemen pada top dengan memeriksa apakah stack kosong dan jika kosong maka menampilkan pesan stack kosong dan jika tidak maka menyimpan top ke temp, mengubah top ke

top->next, dan menghapus temp.

```
#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;
    }
```

7. Method peek() untuk mengembalikan nilai elemen top tanpa menghapusnya dengan cara memeriksa apakah stack kosong dan jika kosong maka menampilkan pesan stack kosong lalu mengembalikan -1 dan jika tidak maka mengembalikan top->data.
8. Method display() untuk menampilkan semua elemen stack dari atas ke bawah dengan memeriksa apakah stack kosong dan jika kosong maka menampilkan pesan stack kosong dan jika tidak maka membuat pointer current dimulai dari top menggunakan perulangan untuk menampilkan data hingga nullptr.

```
    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<<"\n";
    }
};
```

9. Membuat main() dengan menginisialisasi class Stack menjadi s, menambahkan elemen kedalam stack dengan push(), menghapus elemen dari stack menggunakan pop(), dan menampilkan isi stack menggunakan display().

```
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;
};
```

4. Unguided

A. Unguided 1

1. Membuat fungsi boolean isPalindrome untuk menerima sebuah string dalam parameter dan mengembalikan nilai true jika string adalah

palindrom atau false jika bukan, mendefinisikan stack<char> menjadi s untuk menyimpan karakter dari string, dan filteredStr adalah string yang difilter agar hanya mengancung karakter alfanumerik dan diubah menjadi huruf kecil.

2. Membuat filter dengan menggunakan perulangan untuk memeriksa setiap karakter dalam string apakah karakter alfanumerik dan jika benar maka karakter akan dikonversi menjadi huruf kecil dan ditambahkan ke filteredStr.
3. Menambahkan setiap karakter dari filteredStr ke stack s.
4. Memeriksa sifat palindrom dengan melakukan perulangan untuk setiap karakter dari filteredStr dari karakter teratas stack, jika c tidak sama dengan s.top(), maka str bukan palindrom lalu mengembalikan false, dan jika karakter cocok maka karakter teratas dari stack dihapus menggunakan s.pop(), dan jika semua karakter cocok hingga stack habis maka mengembalikan true yang menandakan string adalah palindrom.

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

bool isPalindrome(const string& str) {
    stack<char> s;
    string filteredStr = "";
    for(char c : str) {
        if(isalnum(c)) {
            filteredStr += tolower(c);
        }
    }
    for(char c : filteredStr) {
        s.push(c);
    }
    for(char c : filteredStr) {
        if(c != s.top()) {
            return false;
        }
        s.pop();
    }
    return true;
}
```

5. Membuat main() dengan atribut string input, menampilkan pesan untuk memasukan kalimat, meminta input dengan getLine(cin, input).
6. Memanggil fungsi isPalindrome dengan input menjadi argumen, jika isPalindrome(input) mengembalikan true maka menampilkan pesan kalimat merupakan palindrom, dan jika isPalindrome(input)

mengembalikan false maka menampilkan pesan kalimat bukan palindrom.

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

    if(isPalindrome(input)) {
        cout << "Kalimat tersebut adalah Palindrom" << endl;
    } else {
        cout << "Kalimat tersebut adalah bukan Palindrom" << endl;
    }

    return 0;
}
```

7. Berikut merupakan output dari program tersebut.

```
Masukkan Kalimat: ini
Kalimat tersebut adalah Palindrom
```

8. Program mengabaikan karakter selain huruf dan angka serta mengubah semua huruf menjadi huruf kecil. Ini dilakukan agar perbandingan lebih konsisten, misalnya "Ini" dan "ini" dianggap sama.
9. Setiap karakter dari filteredStr (string yang sudah difilter) dimasukkan ke dalam stack s.
10. Stack menyimpan karakter dalam urutan terbalik karena LIFO (Last In, First Out), artinya karakter pertama yang dimasukkan akan berada paling bawah dan yang terakhir akan berada paling atas.
11. Program kemudian memeriksa setiap karakter di filteredStr dari depan ke belakang dan membandingkannya dengan karakter yang diambil dari stack menggunakan s.top(). Jika ada karakter yang berbeda, program akan mengembalikan false, artinya string bukan palindrom. Jika semua karakter cocok, maka program mengembalikan true, menunjukkan bahwa string adalah palindrom.
12. Berdasarkan hasil dari fungsi isPalindrome, program akan menampilkan apakah input adalah palindrom atau bukan.

B. Unguided 2

1. Membuat fungsi reverseString yang menerima parameter string str yang merupakan kata atau kalimat yang akan dibalik dengan membuat stack karakter s dan menambahkan setiap karakter dalam str kedalam stack menggunakan s.push() dalam perulanga, dan menampilkan karakter – karakter daru stack dalam urutan terbalik menggunakan s,pop() hingga

stack kosong dengan perulangan.

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

void reverseString(const string& str) {
    stack<char> s;
    for(char c : str) {
        s.push(c);
    }
    while(!s.empty()) {
        cout << s.top();
        s.pop();
    }
    cout << " ";
}
```

2. Membuat main() dengan atribut string input, menampilkan pesan untuk memasukan kata, menginput kata menggunakan getLine(cin, input) dan menyimpannya pada word.
3. Memeriksa setiap karakter dalam input dengan perulangan, jika karakter adalah spasi maka kata yang terbentuk akan dibalik menggunakan reverseString dan word direset menjadi string kosong, jika ada kata yang tersisa di word maka program akan membaliknya juga, dan mengembalikan 0.

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

    string word = "";
    cout << "Datastack Array :\nData : ";

    for (char c : input) {
        if (c == ' ') {
            reverseString(word);
            word = "";
        } else {
            word += c;
        }
    }
    if (!word.empty()) {
        reverseString(word);
    }
    cout << endl;
    return 0;
}
```

4. Berikut merupakan output dari program tersebut.

```
Masukkan Kata: Telkom University Purwokerto
Datastack Array :
Data : mokleT ytisrevinU otrekowruP
```

5. Program meminta pengguna memasukkan kalimat.
6. Setiap kata dalam kalimat diproses satu per satu dan dibalik menggunakan stack.
7. Hasilnya ditampilkan sebagai kalimat yang setiap katanya dalam urutan terbalik.

5. Kesimpulan

Stack adalah kumpulan struktur yang didasarkan pada urutan LIFO (Last In First Out) sehingga entitas yang terakhir masuk ke dalam sistem adalah yang pertama keluar dari sistem. Dalam implementasi stack, operasi yang dilakukan dalam bentuk dasarnya antara lain push, yang digunakan untuk menambahkan elemen ke dalam stack, pop yang digunakan untuk mengeluarkan elemen dari stack, dan peek yang digunakan untuk melihat

elemen yang paling atas tanpa menghapusnya. Stack digunakan dalam berbagai skenario, misalnya, perubahan urutan karakter pada kata tertentu, atau dalam penguraian ekspresi, dan logika yang membutuhkan LIFO. Diketahui bahwa stack sangat bermanfaat untuk pemrograman terutama dalam proses seperti operasi memori, rekursi, dan kasus – kasus pengurutan ulang item dengan cara yang cepat.