

**LAPORAN PRAKTIKUM**  
**Modul VII**  
**STACK**



**Disusun Oleh:**  
**Berlian Seva Astryana -2311104067**  
**Kelas**  
**SISE-07-02**

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

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**  
**FAKULTAS INFORMATIKA**  
**TELKOM UNIVERSITY**  
**PURWOKERTO**  
**2024**

## 1. Tujuan

- 1.1 Mampu memahami konsep stack pada struktur data dan algoritma
- 1.2 Mampu mengimplementasikan operasi-operasi pada stack
- 1.3 Mampu memecahkan permasalahan dengan solusi stack

## 2. Landasan Teori

Stack adalah salah satu data yang beroperasi berdasarkan prinsip Last In, First Out (LIFO), dimana elemen terakhir yang ditambahkan akan menjadi elemen pertama yang dihapus. Dalam C++, stack dapat digunakan untuk menyertakan pustaka `<stack>` dan dideklarasikan menggunakan sintaks `stack<type>namaStack`, dengan jenis data yang tidak dapat diubah setelah deklarasi. Elemen-elemen dalam stack tidak diakses melalui indeks tidak diakses melalui indeks, melainkan hanya melalui bagian atas (*top*). Operasi utama yang tersedia pada stack mencakup push untuk menambahkan elemen ke atas tumpukan, pop untuk menghapus elemen teratas, dan top untuk mengakses atau mengubah elemen teratas tanpa menghapusnya. Dengan fungsi-fungsi ini, stack memungkinkan pengelolaan data secara efisien untuk berbagai kebutuhan pemrograman.

Selain operasi dasar, stack juga menyediakan fungsi tambahan seperti `size` untuk mengetahui jumlah elemen dalam stack dan `empty` untuk memeriksa apakah stack kosong. Stack sangat berguna dalam berbagai aplikasi pemrograman, seperti evaluasi ekspresi matematika, manajemen memori, dan pengelolaan panggilan fungsi. Elemen-elemen dalam stack hanya dapat ditambahkan dan dihapus dari bagian atas, menjadikannya solusi ideal untuk situasi di mana data perlu dikelola dengan pola yang terstruktur dan mengikuti prinsip LIFO. Dengan fleksibilitas dan kemudahan penggunaannya, stack menjadi salah satu struktur data fundamental yang sering digunakan dalam pengembangan perangkat lunak.

## 3. Guided

### 3.1 Guided 1

Program:

```
#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;
}

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 element; ";
    s.display();

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

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

    return 0;
}

```

Program tersebut mengimplementasikan struktur data stack, yaitu struktur data yang bekerja dengan prinsip LIFO (Last In, First Out). Dalam stack, elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dihapus. Program ini menggunakan kelas Stack yang memiliki atribut top untuk melacak indeks elemen teratas dan array arr[MAX] untuk menyimpan elemen stack dengan kapasitas maksimum 100. Beberapa operasi utama pada stack adalah push, pop, peek, dan display. Operasi push digunakan untuk menambahkan elemen ke atas stack dengan terlebih dahulu memeriksa apakah stack penuh. Jika stack tidak penuh, elemen ditambahkan, dan indeks top meningkat. Operasi pop menghapus elemen teratas dari stack dengan mengurangi nilai top, setelah memastikan stack tidak kosong. Operasi peek memungkinkan melihat elemen teratas tanpa menghapusnya, sedangkan display digunakan untuk mencetak semua elemen stack dari elemen teratas hingga terbawah.

Pada program utama, beberapa elemen seperti 10, 20, dan 30 ditambahkan ke stack menggunakan operasi push. Kemudian, elemen-elemen tersebut ditampilkan menggunakan display, menunjukkan urutan 30, 20, 10. Setelah itu, operasi pop dilakukan dua kali, menghapus elemen 30 dan 20, sehingga hanya elemen 10 yang tersisa di stack. Proses ini menunjukkan bagaimana stack memproses data berdasarkan prinsip LIFO, di mana elemen yang terakhir masuk akan menjadi elemen pertama yang keluar.

Output:

```
Stack element; 30 20 10
Top element; 30
After popping, stack elements: 10
```

### 3.2 Guided 2

Program:

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

};

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

cout << "Stack element; ";
s.display();

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

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

```

```
cout << "After popping, stack elements: ";  
s.display();  
  
return 0;  
}
```

Program tersebut mengimplementasikan **stack** menggunakan struktur data **linked list**, di mana elemen stack direpresentasikan sebagai node yang saling terhubung. Setiap node memiliki atribut data untuk menyimpan nilai dan pointer next yang menunjuk ke node berikutnya. Elemen teratas stack direpresentasikan oleh pointer top. Operasi utama stack meliputi push, pop, peek, dan display. Operasi push menambahkan elemen baru dengan membuat node baru, menghubungkannya ke elemen teratas saat ini, lalu memperbarui top. Operasi pop menghapus elemen teratas dengan memindahkan pointer top ke elemen berikutnya dan menghapus node sebelumnya dari memori. Operasi peek digunakan untuk melihat nilai elemen teratas tanpa menghapusnya, sedangkan display menampilkan seluruh elemen stack dari atas ke bawah.

Berbeda dengan implementasi sebelumnya yang menggunakan array dengan ukuran tetap, program ini menggunakan linked list sehingga ukuran stack bersifat dinamis dan tidak dibatasi oleh kapasitas tertentu selain memori yang tersedia. Linked list juga menghilangkan risiko **stack overflow** yang mungkin terjadi pada array. Namun, penggunaan linked list membutuhkan memori tambahan untuk pointer pada setiap node, dan operasi iterasi seperti display menjadi lebih lambat dibandingkan array karena elemen diakses secara sekuensial melalui pointer. Implementasi ini lebih fleksibel untuk kebutuhan stack yang tidak terduga ukurannya.

Output:

```
Stack element; 30 20 10  
Top element; 30  
After popping, stack elements: 10
```

#### 4. Unguided

4.1 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.

Contoh:

Kalimat : ini

Kalimat tersebut adalah polindrom

Kalimat : telkom

Kalimat tersebut adalah bukan polindrom

```
Masukan Kalimat : ini  
Kalimat tersebut adalah : Palindrom
```

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

```

using namespace std;

bool isPalindrome(string str) {
    stack<char> s;
    string filtered = "";

    // Filter hanya huruf kecil dan masukkan ke string baru
    for (char c : str) {
        if (isalpha(c)) {
            filtered += tolower(c);
        }
    }

    // Masukkan karakter ke dalam stack
    for (char c : filtered) {
        s.push(c);
    }

    // Periksa apakah string sama saat dibalik
    for (char c : filtered) {
        if (c != s.top()) {
            return false; // Jika ada yang tidak sama
        }
        s.pop();
    }

    return true; // Semua karakter cocok
}

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

    if (isPalindrome(input)) {
        cout << "Kalimat tersebut adalah palindrom.\n";
    } else {
        cout << "Kalimat tersebut bukan palindrom.\n";
    }

    return 0;
}

```

Output:

```

Masukkan kalimat: apa
Kalimat tersebut adalah palindrom.

```

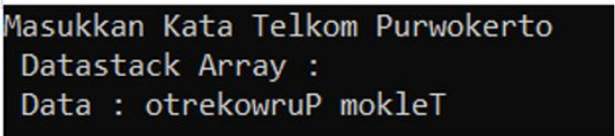
Program ini membalikkan urutan kata dalam sebuah kalimat dengan menggunakan stack. Kalimat yang dimasukkan pengguna dipecah menjadi kata-kata menggunakan objek stringstream, lalu setiap kata dimasukkan ke dalam stack menggunakan operasi push. Karena stack bekerja dengan prinsip LIFO (Last In, First Out), saat kata-kata dikeluarkan dengan operasi pop, urutan kata menjadi terbalik. Jika jumlah kata dalam kalimat kurang dari tiga, program akan menampilkan pesan bahwa kalimat tidak valid. Program ini ringkas karena langsung memproses kalimat menggunakan loop sederhana tanpa fungsi tambahan. Hasil akhirnya adalah kalimat dengan urutan kata terbalik yang ditampilkan di layar.

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

Contoh

Kalimat : Telkom Purwokerto

Hasil : otrekowruP mokleT



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

Program:

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

int main() {
    string input;
    stack<char> s;

    cout << "Masukkan Kata: ";
    getline(cin, input); // Membaca input string dengan spasi

    // Masukkan setiap karakter ke dalam stack
    for (char c : input) {
        s.push(c);
    }

    cout << "Datastack Array:\nData : ";

    // Keluarkan karakter dari stack untuk membalikkan string
    while (!s.empty()) {
        cout << s.top();
        s.pop();
    }

    cout << endl;
    return 0;
}
```



Output:

```
Masukkan Kata: Berlian Seva  
Datastack Array:  
Data : aveS nailreB
```

Program ini menggunakan stack untuk membalikkan urutan karakter dalam kalimat yang diinput oleh pengguna. Ketika pengguna memasukkan sebuah kalimat, setiap karakter, termasuk spasi, dimasukkan ke dalam stack satu per satu menggunakan operasi push. Stack bekerja dengan prinsip LIFO (Last In, First Out), sehingga karakter yang terakhir dimasukkan akan berada di puncak stack. Setelah semua karakter tersimpan di stack, program mulai mengambil karakter dari puncak stack menggunakan operasi pop, yang menghapus karakter tersebut dari stack. Karena karakter diambil dalam urutan terbalik dari saat dimasukkan, program secara otomatis mencetak karakter dalam urutan terbalik, menghasilkan kalimat yang dibalik sepenuhnya.

## 5. Kesimpulan

Praktikum ini membahas implementasi dan aplikasi stack, baik menggunakan array maupun linked list, yang bekerja berdasarkan prinsip LIFO (Last In, First Out). Melalui berbagai program, mahasiswa memahami operasi dasar stack seperti push, pop, dan peek, serta penerapannya untuk memecahkan masalah, seperti pemeriksaan palindrom dan pembalikan kalimat. Praktikum ini juga menunjukkan kelebihan dan kekurangan implementasi stack, serta pentingnya stack sebagai struktur data yang efisien dan fleksibel dalam pengembangan perangkat lunak.

