

# **LAPORAN PRAKTIKUM**

## **MODUL VI STACK**



**Disusun oleh:**  
**Avriel Fitria Rachma Suryantari**  
**NIM: 2311102036**

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

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
PURWOKERTO  
2024**

## **BAB I**

### **TUJUAN PRAKTIKUM**

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

## **BAB II**

### **DASAR TEORI**

Stack adalah salah satu struktur data dasar yang digunakan dalam pemrograman, termasuk C++. Stack beroperasi berdasarkan prinsip LIFO (Last In, First Out), yang berarti elemen terakhir yang dimasukkan ke dalam stack akan menjadi elemen pertama yang dikeluarkan. Dalam C++, stack dapat diimplementasikan menggunakan array atau linked list, tetapi C++ Standard Library menyediakan kelas `std::stack` yang memudahkan penggunaannya. `std::stack` adalah adaptor kontainer yang menyediakan antarmuka standar untuk operasi stack, termasuk `push` untuk menambah elemen ke stack, `pop` untuk menghapus elemen dari stack, `top` untuk mengakses elemen paling atas tanpa menghapusnya, dan `empty` serta `size` untuk memeriksa status stack.

Stack sering digunakan dalam berbagai aplikasi pemrograman seperti manajemen memori, pemrosesan ekspresi aritmatika dan notasi Polandia terbalik (reverse Polish notation), serta pelacakan sejarah tindakan (undo) dalam aplikasi perangkat lunak. Pada C++, penggunaannya melibatkan penanganan objek dan tipe data dengan efisien, memungkinkan pengembangan algoritma yang membutuhkan backtracking, seperti algoritma pencarian depth-first search (DFS). Pemahaman mendalam tentang stack dan implementasinya di C++ sangat penting bagi pengembang untuk memanfaatkan

struktur data ini dalam memecahkan masalah kompleks dan mengembangkan aplikasi yang efisien.

Operasi-operasi umum yang dilakukan pada stack meliputi:

a. Push (Masukkan): Menambahkan elemen ke dalam tumpukan pada posisi paling atas atau ujung.

- Contoh: Jika stack berisi [1, 2, 3] dan kita melakukan push(4), stack akan menjadi [1, 2, 3, 4].

b. Pop (Keluarkan): Menghapus elemen dari posisi paling atas atau ujung tumpukan.

- Contoh: Jika stack berisi [1, 2, 3, 4] dan kita melakukan pop(), stack akan menjadi [1, 2, 3].

c. Top (Atas): Mendapatkan nilai atau melihat elemen teratas pada tumpukan tanpa menghapusnya.

- Contoh: Jika stack berisi [1, 2, 3, 4], top() akan mengembalikan 4, tetapi stack tetap [1, 2, 3, 4].

d. IsEmpty (Kosong): Memeriksa apakah tumpukan kosong atau tidak.

- Contoh: Jika stack berisi [], isEmpty() akan mengembalikan true.

e. IsFull (Penuh): Memeriksa apakah tumpukan penuh atau tidak (terutama pada implementasi tumpukan dengan kapasitas terbatas).

- Contoh: Jika stack berisi [1, 2, 3, 4] dengan kapasitas maksimum 4, isFull() akan mengembalikan true.

f. Size (Ukuran): Mengembalikan jumlah elemen yang ada dalam tumpukan.

- Contoh: Jika stack berisi [1, 2, 3, 4], size() akan mengembalikan 4.

g. Peek (Lihat): Melihat nilai atau elemen pada posisi tertentu dalam tumpukan tanpa menghapusnya.

- Contoh: Jika stack berisi [1, 2, 3, 4] dan kita melakukan peek(2), akan mengembalikan nilai 2 tanpa mengubah stack.

h. Clear (Hapus Semua): Mengosongkan atau menghapus semua elemen dari tumpukan.

- Contoh: Jika stack berisi [1, 2, 3, 4], clear() akan mengubahnya menjadi [].

i. Search (Cari): Mencari keberadaan elemen tertentu dalam tumpukan.

- Contoh: Jika stack berisi [1, 2, 3, 4] dan kita melakukan search(3), akan mengembalikan true jika 3 ada dalam stack.

Operasi-operasi ini memungkinkan pengguna untuk memanipulasi data dalam stack secara efisien, sesuai dengan kebutuhan spesifik dari berbagai aplikasi dan algoritma.

## BAB III

## GUIDED

### 1. Guided 1

#### Source code

```
#include <iostream>
using namespace std;
string arrayBuku[5];
int maksimal = 5, top = 0;
bool isFull()
{
    return (top == maksimal);
}
bool isEmpty()
{
    return (top == 0);
}
void pushArrayBuku(string data)
{
    if (isFull())
    {
        cout << "Data telah penuh" << endl;
    }
    else
    {
        arrayBuku[top] = data;
        top++;
    }
}
void popArrayBuku()
{
    if (isEmpty())
    {
        cout << "Tidak ada data yang dihapus" << endl;
```

```

    }
    else
    {
        arrayBuku[top - 1] = "";
        top--;
    }
}

void peekArrayBuku(int posisi)
{
    if (isEmpty())
    {
        cout << "Tidak ada data yang bisa dilihat" << endl;
    }
    else
    {
        int index = top;
        for (int i = 1; i <= posisi; i++)
        {

            index--;

        }
        cout << "Posisi ke " << posisi << " adalah " <<
arrayBuku[index] << endl;
    }
}

int countStack()
{
    return top;
}

void changeArrayBuku(int posisi, string data)
{
    if (posisi > top)
    {
        cout << "Posisi melebihi data yang ada" << endl;
    }
}

```

```

    }
    else
    {
        int index = top;
        for (int i = 1; i <= posisi; i++)
        {
            index--;
        }
        arrayBuku[index] = data;
    }
}

void destroyArraybuku()
{
    for (int i = top; i >= 0; i--)
    {
        arrayBuku[i] = "";
    }
    top = 0;
}

void cetakArrayBuku()
{
    if (isEmpty())
    {
        cout << "Tidak ada data yang dicetak" << endl;
    }
    else
    {
        for (int i = top - 1; i >= 0; i--)
        {
            cout << arrayBuku[i] << endl;
        }
    }
}

int main()

```

```
{  
    pushArrayBuku("Kalkulus");  
    pushArrayBuku("Struktur Data");  
    pushArrayBuku("Matematika Diskrit");  
    pushArrayBuku("Dasar Multimedia");  
    pushArrayBuku("Inggris");  
    cetakArrayBuku();  
    cout << "\n";  
    cout << "Apakah data stack penuh? " << isFull() << endl;  
    cout << "Apakah data stack kosong? " << isEmpty() << endl;  
    peekArrayBuku(2);  
    popArrayBuku();  
    cout << "Banyaknya data = " << countStack() << endl;  
    changeArrayBuku(2, "Bahasa Jerman");  
    cetakArrayBuku();  
    cout << "\n";  
    destroyArraybuku();  
    cout << "Jumlah data setelah dihapus: " << top << endl;  
    cetakArrayBuku();  
    return 0;  
}
```



### Screenshoot program

```
PS C:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul6\output> & .\'guided1.exe'
Inggris
Dasar Multimedia
Matematika Diskrit
Struktur Data
Kalkulus

Apakah data stack penuh? 1
Apakah data stack kosong? 0
Posisi ke 2 adalah Dasar Multimedia
Banyaknya data = 4
Dasar Multimedia
Bahasa Jerman
Struktur Data
Kalkulus

Jumlah data setelah dihapus: 0
Tidak ada data yang dicetak
PS C:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul6\output> █
```

### Deskripsi program

Program ini mengimplementasikan stack sederhana menggunakan array untuk menyimpan dan mengelola data buku. Dengan kapasitas maksimal 5 buku, program menyediakan fungsi untuk menambah buku (`pushArrayBuku`), menghapus buku (`popArrayBuku`), melihat buku di posisi tertentu (`peekArrayBuku`), menghitung jumlah buku dalam stack (`countStack`), mengubah data buku di posisi tertentu (`changeArrayBuku`), mengosongkan seluruh stack (`destroyArraybuku`), dan mencetak seluruh buku dalam stack (`cetakArrayBuku`). Fungsi `isFull` dan `isEmpty` digunakan untuk memeriksa apakah stack penuh atau

kosong. Program ini juga mendemonstrasikan penggunaan fungsi-fungsi tersebut dalam fungsi main dengan menambah beberapa buku, mencetak buku-buku tersebut, memeriksa kondisi stack, mengubah data buku, dan akhirnya mengosongkan stack serta mencetak ulang isi stack setelah dihapus.

## LATIHAN KELAS - 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.

### Source code

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

using namespace std;

bool isPalindrome(const string& str) {
    stack<char> charStack;
    string cleanStr;

    for (char c : str) {
        if (isalnum(c)) {
            cleanStr += tolower(c);
            charStack.push(tolower(c));
        }
    }

    for (char c : cleanStr) {
        if (charStack.top() != c) {
            return false;
        }
        charStack.pop();
    }

    return true;
}
```

```
int main() {
    char choice;
    string sentence;

    do {
        cout << "\nMasukkan Kalimat: ";
        getline(cin, sentence);

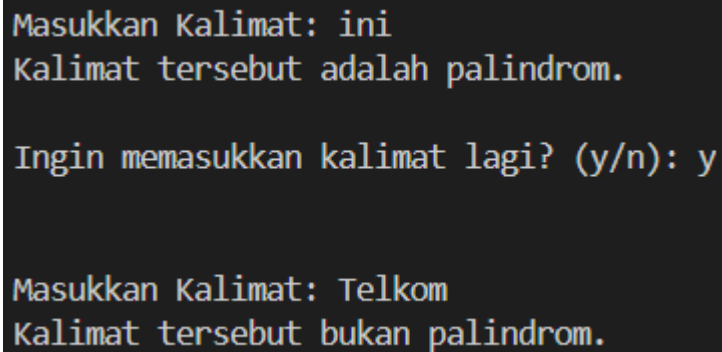
        if (isPalindrome(sentence)) {
            cout << "Kalimat tersebut adalah palindrom." << endl;
        } else {
            cout << "Kalimat tersebut bukan palindrom." << endl;
        }

        cout << "\nIngin memasukkan kalimat lagi? (y/n): ";
        cin >> choice;
        cin.ignore();

    } while (tolower(choice) == 'y');

    return 0;
} //Ilhan Sahal M // 2311102029
```

## Screenshot Program

A screenshot of a terminal window with a dark background and light-colored text. The text shows the program's execution flow: it prompts for a word, checks if 'ini' is a palindrome (yes), asks if the user wants to continue (y/n), and then checks if 'Telkom' is a palindrome (no).

```
Masukkan Kalimat: ini
Kalimat tersebut adalah palindrom.

Ingin memasukkan kalimat lagi? (y/n): y

Masukkan Kalimat: Telkom
Kalimat tersebut bukan palindrom.
```

## Deskripsi Program

Program ini adalah implementasi sederhana untuk memeriksa apakah sebuah kalimat yang dimasukkan oleh pengguna merupakan palindrom atau tidak. Sebuah kalimat dikatakan palindrom jika urutan karakternya sama baik dari depan maupun dari belakang, tanpa memperhatikan spasi, kapitalisasi, atau tanda baca. Dalam program ini, sebuah stack digunakan untuk menyimpan karakter-karakter yang relevan dari kalimat yang dimasukkan. Karakter-karakter tersebut kemudian dibandingkan dengan karakter-karakter dalam bentuk "bersih" dari kalimat tersebut, yang mana hanya berisi karakter alfanumerik dan diubah menjadi huruf kecil. Jika kedua urutan karakter ini sama, maka kalimat tersebut dianggap sebagai palindrom. Program juga memberikan opsi kepada pengguna untuk memasukkan kalimat baru atau mengakhiri program setelah setiap pengecekan.

Meskipun sederhana, program ini memanfaatkan konsep stack untuk melakukan pengecekan palindrom dengan efisien. Penggunaan stack memungkinkan program untuk membandingkan karakter-karakter dari awal dan akhir kalimat dengan mudah, menghindari perluasan algoritma yang lebih rumit. Dengan memanfaatkan fitur-fitur dari C++ seperti `std::stack` dan fungsi-fungsi dari `<string>` dan `<cctype>`, program ini memberikan solusi yang efektif dan mudah dipahami bagi pengguna untuk memeriksa apakah kalimat yang dimasukkan merupakan palindrom.

**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?

### Source Code

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

using namespace std;

string reverseSentence(string sentence) {
    stack<char> charStack;
    string reversedSentence = "";

    for (char &c : sentence) {
        if (c != ' ') {
            charStack.push(c);
        } else {
            while (!charStack.empty()) {
                reversedSentence += charStack.top();
                charStack.pop();
            }
            reversedSentence += ' ';
        }
    }

    while (!charStack.empty()) {
        reversedSentence += charStack.top();
        charStack.pop();
    }

    return reversedSentence;
}

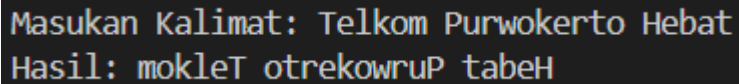
int main() {
    string kalimat, hasil;
    cout << "Masukan Kalimat: ";
    getline(cin, kalimat);

    hasil = reverseSentence(kalimat);

    cout << "Hasil: " << hasil << endl;

    return 0;
} //Ilhan Sahal Mansiz // 2311102029
```

## Screenshot Program



Masukan Kalimat: Telkom Purwokerto Hebat  
Hasil: mokleT otrekowruP tabeH

## Deskripsi Program

Program ini merupakan implementasi sederhana dari sebuah fungsi untuk membalikkan urutan setiap kata dalam sebuah kalimat. Ketika pengguna memasukkan sebuah kalimat, program akan memprosesnya dengan mengubah urutan setiap kata dalam kalimat tersebut menjadi urutan yang terbalik. Langkah-langkahnya dimulai dengan membagi kalimat menjadi kata-kata dan menyimpan setiap karakter dari kata-kata tersebut ke dalam sebuah stack. Kemudian, stack tersebut akan digunakan untuk membalikkan urutan setiap karakter dari setiap kata, dan hasilnya akan disalin ke dalam sebuah string yang menyimpan kalimat yang telah dibalik. Proses ini dilakukan dengan memindahkan karakter-karakter dari stack ke string hasil dalam urutan terbalik. Dengan demikian, output program ini akan menampilkan kalimat yang sama dengan inputnya, namun dengan urutan setiap kata telah dibalik.

Meskipun program ini sederhana, ia memberikan contoh penerapan struktur data stack dalam pemrosesan teks. Penggunaan stack memungkinkan program untuk secara efisien membalikkan urutan setiap kata dalam sebuah kalimat dengan mempertahankan urutan kata-kata tersebut. Penggunaan fitur-fitur dasar dari C++ seperti `<stack>` dan `<string>` memungkinkan program ini untuk memberikan solusi yang efektif tanpa memerlukan implementasi yang rumit. Dengan demikian, program ini dapat digunakan sebagai referensi untuk pemrosesan teks yang memerlukan pembalikan urutan kata-kata secara sederhana namun efisien.

Output dari program ini akan membalikkan urutan setiap kata dalam kalimat yang dimasukkan oleh pengguna. Setiap kata dalam kalimat akan dibalikkan urutannya, tetapi urutan kata-kata tersebut dalam kalimat akan tetap sama. Misalnya, jika pengguna memasukkan kalimat "Ini adalah contoh kalimat", outputnya akan menjadi "ini salada ohntoc tamilak". Ini dilakukan dengan menggunakan fungsi `reverseSentence` yang dibuat dalam program.

## **BAB IV**

### **KESIMPULAN**

Modul 6 tentang stack pada C++ menawarkan pemahaman yang mendalam tentang struktur data stack dan implementasinya dalam pemrograman. Stack adalah struktur data yang fundamental dalam komputer, beroperasi berdasarkan prinsip LIFO (Last In, First Out), di mana elemen terakhir yang dimasukkan ke dalam stack akan menjadi elemen pertama yang dikeluarkan. Dalam modul ini, kita mempelajari konsep dasar stack, termasuk operasi-operasi umum seperti push, pop, top, isEmpty, dan isFull, serta implementasinya dalam bahasa pemrograman C++. Modul ini juga mengeksplorasi berbagai aplikasi praktis dari stack dalam pemrograman, termasuk penyelesaian masalah dengan menggunakan stack seperti pengecekan palindrom, pengelolaan memori, dan algoritma pencarian.

## **BAB V**

### **REFERENSI**

- <https://www.nblognlife.com/2014/04/stack-pada-c.html>
- <https://glints.com/id/lowongan/stack-adalah/>
- <https://www.trivusi.web.id/2022/07/struktur-data-stack.html>