

LAPORAN PRAKTIKUM

Modul 7

“Stack”



Disusun Oleh:

Fahmi Hasan Asagaf -2311104074

SE 07 02

Dosen :

Wahyu Andi Saputra

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. Tujuan

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

2. Landasan Teori

1. Pengertian Stack

Stack adalah salah satu struktur data yang menerapkan prinsip LIFO (Last In, First Out).

Ini berarti bahwa elemen terakhir yang dimasukkan ke dalam stack adalah elemen pertama yang akan dikeluarkan. Struktur data ini sering digunakan dalam pemrograman untuk menyelesaikan berbagai masalah, seperti pengelolaan fungsi rekursif, evaluasi ekspresi aritmatika, dan pengelolaan data sementara.

2. Karakteristik Stack

1. LIFO: Elemen terakhir yang dimasukkan adalah elemen pertama yang dikeluarkan.
2. Operasi yang umum dilakukan:
 - Push: Menambahkan elemen baru ke puncak stack.
 - Pop: Menghapus elemen dari puncak stack.
 - Top/Peak: Mengakses elemen di puncak stack tanpa menghapusnya.
 - isEmpty: Mengecek apakah stack kosong.
 - isFull: Mengecek apakah stack penuh (bergantung pada implementasi).

3. Implementasi Stack di C++

Stack dapat diimplementasikan dengan dua cara utama:

1. Array-Based Stack: Menggunakan array sebagai dasar penyimpanan.
2. Linked List-Based Stack: Menggunakan linked list untuk mendukung operasi push dan pop.

Selain itu, C++ menyediakan implementasi stack bawaan melalui library Standard Template Library (STL).

3. Guided

Guided 1

```
#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" << endl;
            return;
        }
        arr[++top] = x;
    }

    int pop() {
        if (isEmpty()) {
            cout << "Stack Underflow" << endl;
            return -1;
        }
        return arr[top--];
    }
}
```

```
int peek() {
    if (isEmpty()) {
        cout << "Stack Underflow" << endl;
        return -1;
    }
    return arr[top];
}

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

    return 0;
}
```

Screenshot output

```
Stack elements: 30 20 10  
Top element: 30
```

```
Process returned 0 (0x0)   execution time : 0.267 s  
Press any key to continue.  
|
```

Guided2

```
#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()) {
        cout << "Stack is empty\n";
        return -1; // Atau bisa return nilai lain untuk menandakan stack kosong
    }
    return top->data;
}

void display() {
    if (isEmpty()) {
        cout << "Stack is empty\n";
        return;
    }
    Node* current = top;
    while (current != nullptr) {
        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() << endl;  
  
    s.pop();  
  
    cout << "Stack after pop: ";  
    s.display();  
  
    return 0;  
}
```

Screenshot output

```
Stack elements: 30 20 10  
Top element: 30  
Stack after pop: 20 10  
  
Process returned 0 (0x0)   execution time : 0.067 s  
Press any key to continue.  
|
```

4. Unguided

Unguided 1

```
#include <iostream>
#include <stack>
#include <algorithm> // untuk transform
#include <cctype>    // untuk isalnum

using namespace std;

bool isPalindrome(string input) {
    // Menghapus spasi dan simbol, lalu mengubah menjadi huruf kecil
    string cleanedInput = "";
    for (char ch : input) {
        if (isalnum(ch)) { // Memeriksa apakah karakter adalah huruf/angka
            cleanedInput += tolower(ch);
        }
    }

    // Menggunakan stack untuk membalik string
    stack<char> s;
    for (char ch : cleanedInput) {
        s.push(ch);
    }

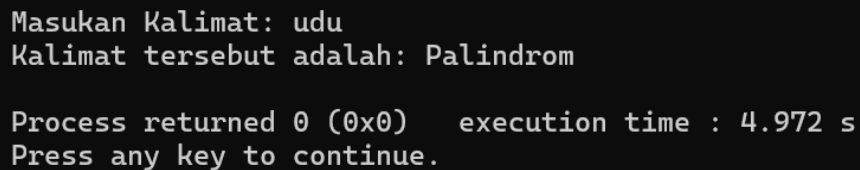
    string reversed = "";
    while (!s.empty()) {
        reversed += s.top();
        s.pop();
    }

    // Membandingkan string asli yang dibersihkan dengan string terbalik
    return cleanedInput == reversed;
}
```



```
int main() {  
    string input;  
    cout << "Masukan Kalimat: ";  
    getline(cin, input);  
  
    if (isPalindrome(input)) {  
        cout << "Kalimat tersebut adalah: Palindrom" << endl;  
    } else {  
        cout << "Kalimat tersebut adalah: Bukan Palindrom" << endl;  
    }  
  
    return 0;  
}
```

Screenshot output



```
Masukan Kalimat: udu  
Kalimat tersebut adalah: Palindrom  
  
Process returned 0 (0x0)   execution time : 4.972 s  
Press any key to continue.  
|
```

Penjelasan Cara Kerja Program:

1. Input dan Normalisasi:

Program meminta pengguna memasukkan kalimat.

Karakter yang bukan huruf atau angka dihapus.

Semua huruf diubah menjadi huruf kecil menggunakan fungsi `tolower`.

2. Menggunakan Stack:

Karakter dari string yang sudah dinormalisasi dimasukkan ke dalam stack.

Karena stack menggunakan prinsip *Last In First Out* (LIFO), string bisa dibalik dengan mengambil karakter dari stack.

3. Pemeriksaan Palindrom:

String asli yang sudah dibersihkan dibandingkan dengan versi terbaliknya.

Jika sama, maka kalimat adalah palindrom. Jika tidak, maka bukan.

4. Output:

Program akan menampilkan hasil apakah kalimat tersebut palindrom atau bukan.

Unguided 2

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

class Stack {
private:
    char stack[100]; // Array untuk menyimpan data stack
    int top;         // Pointer ke elemen teratas stack

public:
    Stack() {
        top = -1; // Inisialisasi stack kosong
    }

    bool isFull() {
        return top == 99; // Maksimal kapasitas 100 elemen
    }

    bool isEmpty() {
        return top == -1;
    }

    void push(char ch) {
        if (!isFull()) {
            stack[++top] = ch; // Masukkan elemen ke stack
        } else {
            cout << "Stack penuh!\n";
        }
    }
}
```

```
char pop() {
    if (!isEmpty()) {
        return stack[top--]; // Hapus elemen dari stack
    } else {
        cout << "Stack kosong!\n";
        return '\0';
    }
}

};

void reverseString(const string& input) {
    Stack stack;
    cout << "Data Stack Array:\n";

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

    // Tampilkan karakter dari stack dalam urutan terbalik
    cout << "Data: ";
    while (!stack.isEmpty()) {
        cout << stack.pop();
    }
    cout << endl;
}

int main() {
    string sentence;

    cout << "Masukkan kata: ";
    getline(cin, sentence);

    reverseString(sentence);

    return 0;
}
```

Screenshot Output

```
Masukkan kata: Telkom Bandung
Data Stack Array:
Data: gnudnaB mokleT

Process returned 0 (0x0)   execution time : 6.791 s
Press any key to continue.
|
```

Deskripsi program

Operasi Stack:

push: Menyimpan karakter ke dalam stack.

pop: Menghapus dan mengembalikan karakter teratas dari stack.

String Operations:

getline(cin, sentence): Membaca input berupa string dari pengguna.

Karakteristik LIFO Stack:

Data terakhir yang dimasukkan ke stack akan keluar lebih dulu, menghasilkan pembalikan urutan karakter.

5. Kesimpulan

Stack adalah salah satu struktur data yang bekerja dengan prinsip Last In, First Out (LIFO), di mana data yang dimasukkan terakhir akan dikeluarkan pertama. Dalam praktikum ini, implementasi stack menggunakan C++ menunjukkan cara kerja operasi dasar seperti `push` untuk menambahkan elemen dan `pop` untuk menghapus elemen. Stack sangat berguna untuk berbagai kasus, seperti membalikkan string, mengevaluasi ekspresi, atau menyimpan data sementara dalam proses rekursif. Melalui implementasi ini, mahasiswa dapat memahami pentingnya stack dalam pengelolaan data secara efisien, meskipun perlu diperhatikan masalah seperti overflow atau underflow jika kapasitas stack terbatas.