

LAPORAN PRAKTIKUM STRUKTUR DATA

PERTEMUAN 7

STACK



Nama :

Reyner Atira Prasetyo (2311104057)

S1SE-07-02

Dosen :

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

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

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

II. TOOL

- Visual Studio Code
- GCC

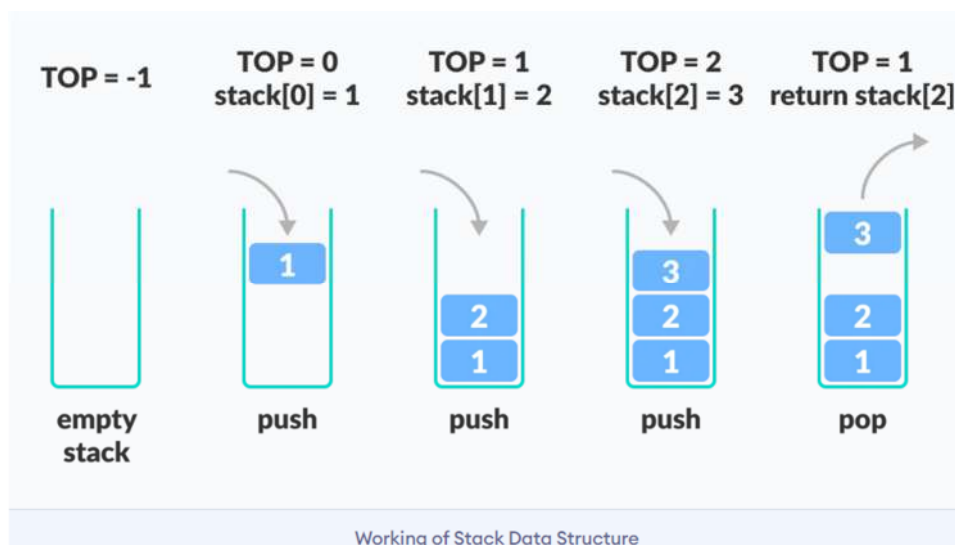
III. DASAR TEORI

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..



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

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

IV. GUIDED

1. guided.cpp

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

using namespace std;

class Stack
{
    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";
            return;
        }
        arr[++top] = x;
    }

    void pop() {
        if (isEmpty())
        {
            cout << "Stack Underflow";
            return;
        }
        top--;
    }

    int peek() {
        if (!isEmpty())
        {
            return arr[top];
        }
    }
}
```

```

        cout << "Stack Underflow";
        return 0;
    }

    void display() {
        if (isEmpty())
        {
            cout << "Stack is Empty";
            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 are: ";
    s.display();

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

    s.pop();
    s.pop();
    cout << "After popping, the stack is ";
    s.display();

    return 0;
}

```

Output :

```

Stack elements are: 30 20 10
Top element is: 30
After popping, the stack is 10
PS D:\PRAKTIKUM\Struktur Data\pertemuan7>

```

2. guided2.cpp

```

#include <iostream>

using namespace std;

class Node
{
public:
    int data;
    Node *next;
    Node(int value)
    {

```

```

        data = value;
        next = NULL;
    }
};

class Stack {
private:
    Node *top;

public:
    Stack() { top = NULL; }
    bool isEmpty() { return top == NULL; }
    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 Underflow\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 elements are: ";
    s.display();

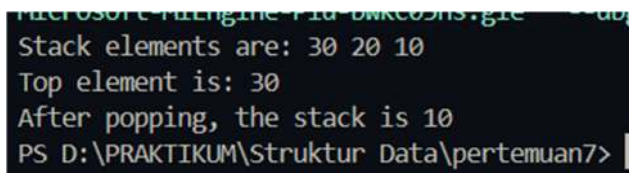
    cout << "Top element is: " << s.peek() << "\n";

    s.pop();
    s.pop();
    cout << "After popping, the stack is ";
    s.display();

    return 0;
}

```

Output :



```

Stack elements are: 30 20 10
Top element is: 30
After popping, the stack is 10
PS D:\PRAKTIKUM\Struktur Data\pertemuan7>

```

V. UNGUIDED

1. palindrome.cpp

```

#include <iostream>
#include <stack>
#include <string>
#include <cctype> // Untuk isalnum dan tolower
using namespace std;

/**
 * @brief Fungsi untuk memeriksa apakah kalimat adalah palindrom
 *
 * This function filters out non-alphanumeric characters from the input string
 and converts
 * all characters to lowercase. It then uses a stack to compare the filtered
 string with its
 * reverse to determine if it is a palindrome.
 *
 * @param str The input string to be checked.
 * @return true if the input string is a palindrome, false otherwise.
 */
bool isPalindrome(string str) {
    stack<char> charStack;
    string filtered = "";

    // Filter karakter non-alfanumerik dan simpan dalam lowercase
    for (char ch : str) {
        if (isalnum(ch)) {
            char lowerChar = tolower(ch);
            filtered += lowerChar;
            charStack.push(lowerChar); // Masukkan ke stack
        }
    }
}

```

```

// Bandingkan string yang difilter dengan isi stack
for (char ch : filtered) {
    if (ch != charStack.top()) {
        return false; // Bukan palindrom
    }
    charStack.pop(); // Hapus elemen teratas stack
}

return true; // Palindrom jika semua cocok
}

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

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

    return 0;
}

```

Penjelasan :

1. Penyaringan Karakter:

- Program memfilter kalimat input sehingga hanya karakter alfanumerik (huruf dan angka) yang digunakan.
- Karakter diubah menjadi huruf kecil agar perbandingan tidak sensitif terhadap kapitalisasi.

2. Penyimpanan di Stack:

- Setiap karakter yang difilter dimasukkan ke dalam stack. Stack adalah struktur data LIFO (Last In First Out), sehingga elemen terakhir yang masuk akan keluar lebih dulu.

3. Pengecekan Palindrom:

- Program membandingkan setiap karakter dalam string yang telah difilter dengan karakter yang diambil dari stack.
- Jika semua karakter cocok (dari awal hingga akhir), maka kalimat tersebut adalah palindrom.

4. Hasil Akhir:

- Jika perbandingan tidak cocok pada satu karakter pun, program akan langsung menyatakan kalimat bukan palindrom.

- Jika semua cocok, maka kalimat adalah palindrom

Output :

Saat palindrome :

```
Microsoft-MIEngine-Pid-tuootrww.fgs' '--dt
Masukkan sebuah kalimat: ini
Kalimat tersebut adalah palindrom.
PS D:\PRAKTIKUM\Struktur Data\pertemuan7>
```

Saat bukan palindrome :

```
Microsoft-MIEngine-Pid-0ioa1xhs.1ct' '--dt
Masukkan sebuah kalimat: telkom
Kalimat tersebut bukan palindrom.
PS D:\PRAKTIKUM\Struktur Data\pertemuan7>
```

2. balikkata.cpp

```
#include <iostream>
#include <stack>
#include <sstream> // Untuk memisahkan kata dalam string
#include <algorithm> // Untuk fungsi reverse
using namespace std;

/**
 * @brief Fungsi untuk membalikkan huruf dalam kata.
 *
 * This function takes a string as input and returns a new string
 * with the characters in reverse order.
 *
 * @param word The string to be reversed.
 * @return A new string with the characters of the input word in reverse
 * order.
 */
string reverseWord(string word) {
    reverse(word.begin(), word.end()); // Membalikkan huruf dalam kata
    return word;
}

/**
 * @brief Fungsi untuk membalikkan kalimat dan kata-kata di dalamnya.
 *
 * This function takes a sentence as input, splits it into individual words,
 * reverses the characters
 * in each word, and then reverses the order of the words in the sentence.
 *
 * @param sentence The input sentence to be reversed.
 * @return A string containing the sentence with the order of words and
 * characters in each word reversed.
 */
string reverseSentence(string sentence) {
    stack<string> wordStack; // Stack untuk menyimpan kata-kata
    stringstream ss(sentence); // Untuk memisahkan kata-kata
    string word, reversed = "";
```



```

    // Memasukkan kata-kata ke dalam stack setelah membalikkan huruf dalam
    setiap kata
    while (ss >> word) {
        wordStack.push(reverseWord(word)); // Balikkan huruf dalam kata dan
        simpan di stack
    }

    // Mengambil kata-kata dari stack dan membentuk kalimat terbalik
    while (!wordStack.empty()) {
        reversed += wordStack.top(); // Ambil kata teratas dari stack
        wordStack.pop(); // Hapus kata dari stack
        if (!wordStack.empty()) reversed += " "; // Tambahkan spasi jika masih
        ada kata
    }

    return reversed;
}

int main() {
    string input;
    cout << "Masukkan sebuah kalimat (minimal 3 kata): ";
    getline(cin, input);

    // Validasi panjang kalimat
    stringstream ss(input);
    int wordCount = 0;
    string temp;
    while (ss >> temp) {
        wordCount++;
    }

    if (wordCount < 3) {
        cout << "Kalimat harus memiliki minimal 3 kata!\n";
        return 1;
    }

    // Membalikkan kalimat dan huruf-huruf dalam kata
    string result = reverseSentence(input);
    cout << "Kalimat setelah dibalik: " << result << endl;

    return 0;
}

```

Penjelasan :

1. Header Files:

- `#include <iostream>`: Untuk input/output ke layar.
- `#include <stack>`: Untuk menggunakan struktur data stack.
- `#include <sstream>`: Untuk memisahkan kalimat menjadi kata-kata.
- `#include <algorithm>`: Untuk menggunakan fungsi reverse dalam membalikkan kata.

2. Fungsi reverseWord(string word):

- Membalikkan urutan huruf dalam satu kata.
- Fungsi ini menggunakan `std::reverse(word.begin(), word.end())` untuk membalikkan urutan karakter dalam string.

3. Fungsi reverseSentence(string sentence):

- Menggunakan stack untuk membalikkan urutan kata dalam kalimat dan membalikkan huruf dalam setiap kata.
- Proses:
 - Kalimat dipisahkan menjadi kata-kata menggunakan stringstream.
 - Setiap kata dibalik hurufnya dengan memanggil reverseWord dan kemudian dimasukkan ke dalam stack.
 - Stack diproses dengan mengambil kata teratas dan membentuk kalimat terbalik. Spasi ditambahkan di antara kata-kata.

4. Fungsi main():

- Input: Meminta pengguna untuk memasukkan kalimat minimal 3 kata.
- Validasi: Mengecek jumlah kata dalam kalimat menggunakan stringstream. Jika jumlah kata kurang dari 3, program berhenti.
- Proses: Memanggil reverseSentence untuk membalikkan kalimat dan kata-kata.
- Output: Menampilkan hasil kalimat yang sudah dibalik.

VI. KESIMPULAN

Praktikum ini membahas materi mengenai struktur data Stack, meliputi penjelasan konsep dasar, metode, dan prosedur yang terkait seperti push, pop, dan peek, serta penerapannya dalam pemrograman C++. Stack, sebagai struktur data LIFO (Last In, First Out), digunakan untuk menyimpan elemen di mana elemen terakhir yang dimasukkan adalah elemen pertama yang dikeluarkan. Implementasi dalam C++ mencakup pembuatan program dengan fungsi-fungsi untuk menambah elemen (push), menghapus elemen (pop), dan melihat elemen puncak (peek). Praktikum ini menunjukkan bagaimana struktur data Stack dapat digunakan dalam berbagai aplikasi, seperti pemrosesan undo/redo, evaluasi ekspresi matematika, dan simulasi panggilan fungsi. Hasil dari praktikum ini membantu memahami penerapan Stack secara efektif dalam penyelesaian masalah komputasi.