

LAPORAN PRAKTIKUM

MODUL 6 STACK



**Disusun oleh:
Bayu Kuncoro Adi
NIM: 2311102031**

**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

A. TUJUAN PRAKTIKUM

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

BAB II

DASAR TEORI

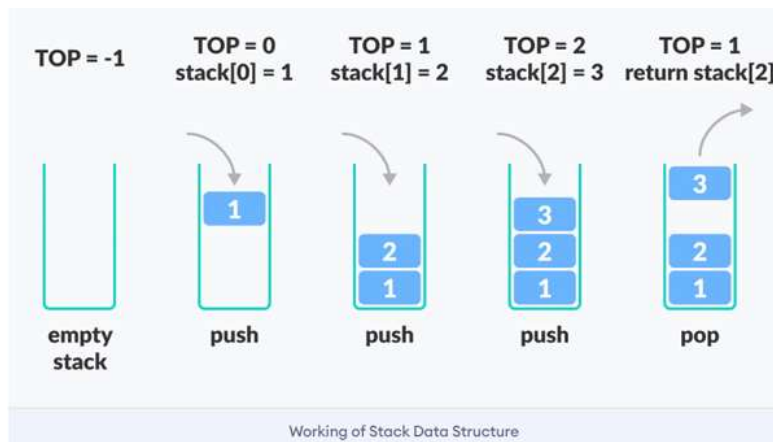
A. DASAR TEORI

1. Pengertian Stack

Stack adalah seperti tumpukan piring di kafetaria; bayangkan saat Anda menumpuk piring bersih satu per satu. Yang Anda letakkan pertama kali akan berada di dasar, dan yang terakhir akan berada di atas. Ketika seseorang mengambil piring, yang diambil adalah yang teratas, bukan yang ada di bawahnya. Konsep ini, di mana elemen terakhir yang dimasukkan adalah yang pertama dikeluarkan, menjadi dasar dari tumpukan. Jadi, secara sederhana, stack adalah cara penyimpanan data di mana yang terakhir datang adalah yang pertama dilayani, mirip dengan saat Anda menumpuk piring; piring teratas adalah yang akan Anda ambil terlebih dahulu. Dalam pemrograman C++, stack merupakan salah satu struktur data yang sangat berguna dan seringkali diimplementasikan menggunakan array atau linked list. Bayangkan stack seperti tumpukan buku di meja; setiap kali Anda menambahkan buku baru, itu diletakkan di atas buku-buku yang sudah ada. Operasi utama dalam stack adalah push untuk menambahkan buku baru ke tumpukan dan pop untuk mengambil buku paling atas dari tumpukan. Ada juga operasi lain seperti peek untuk sekadar melihat buku paling atas tanpa mengambilnya, dan isEmpty untuk memeriksa apakah tumpukan kosong atau tidak.

Penerapan stack dalam pemrograman sangat beragam, mulai dari penyelesaian masalah matematika hingga manajemen memori dalam sistem komputer. Karakteristik LIFO (Last In, First Out) membuat stack sangat berguna untuk mempertahankan urutan operasi atau data yang penting dan memudahkan akses ke data terbaru yang dimasukkan. Namun, ada juga dua kondisi yang perlu diwaspadai dalam struktur data stack, yaitu underflow dan overflow, berikut penjelasannya:

- Stack underflow terjadi saat kita mencoba mengambil buku dari tumpukan yang kosong, seperti mencoba mengambil buku dari tumpukan yang tidak memiliki buku sama sekali.
- Stack overflow terjadi saat tumpukan sudah penuh dengan buku namun kita masih mencoba menambahkan buku baru ke dalamnya, seperti mencoba menambahkan buku baru ke tumpukan yang sudah terlalu tinggi untuk menampungnya.



2. Jenis- jenis operasi pada Stack:

Operasi pada stack melibatkan serangkaian fungsi yang memberikan kita kemampuan untuk berinteraksi dengan struktur data ini. Mari kita jabarkan beberapa operasi dasar yang dapat kita lakukan pada stack:

- Push (Masukkan):** Bayangkan menambahkan satu per satu buku ke tumpukan yang sudah Anda susun. Setiap buku yang Anda tambahkan diletakkan di atas tumpukan, sehingga semakin tinggi tumpukan tersebut.
- Pop (Keluarkan):** Sekarang, bayangkan mengambil buku dari atas tumpukan yang telah Anda susun. Ketika Anda mengambil buku, yang Anda ambil adalah buku paling atas, yang paling terlihat.
- Top (Atas):** Ketika Anda ingin melihat judul buku teratas di tumpukan tanpa mengganggu susunannya.
- IsEmpty (Kosong):** Anda mungkin ingin memeriksa apakah tumpukan buku kosong atau masih ada buku yang tersisa di dalamnya.
- IsFull (Penuh):** Namun, jika Anda hanya memiliki sedikit ruang untuk menambahkan buku baru, Anda perlu memeriksa apakah tumpukan sudah penuh atau masih ada ruang kosong.
- Size (Ukuran):** Ketika Anda ingin tahu berapa banyak buku yang telah Anda susun di tumpukan tersebut.
- Peek (Lihat):** Jika Anda ingin melihat buku tertentu di dalam tumpukan tanpa harus mengambilnya dari susunannya.
- Clear (Hapus Semua):** Ketika tiba saatnya untuk membersihkan tumpukan, Anda ingin mengosongkan semua buku dari tumpukan.
- Search (Cari):** Dan terakhir, jika Anda ingin menemukan apakah buku tertentu ada di dalam tumpukan, Anda bisa mencari dengan cepat.

3. Kelebihan dan kekurangan dalam menggunakan Stack:

a. Kelebihan

- Menggunakan metode LIFO untuk membantu mengelola data dengan mudah dan efektif.
- Secara otomatis membersihkan objek yang tidak lagi diperlukan.
- Tidak mudah rusak karena ukuran variabel yang tetap.
- Ukuran variabel tidak dapat diubah.
- Mengontrol memori secara mandiri.

b. Kekurangan

- Memori stack cenderung terbatas.
- Ada kemungkinan stack akan meluap atau overflow jika objek terlalu banyak dimasukan.
- Tidak dapat mengakses data secara acak, karena harus mengeluarkan tumpukan paling atas terlebih dahulu untuk membuat proses pencarian menjadi lebih terstruktur dan berurutan.

BAB III

GUIDED

1. Guided 1

Program:

```
#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 bisa 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;
cout << "\n";

```

```

peekArrayBuku(2);

popArrayBuku();
cout << "\n";

cout << "Banyaknya data = " << countStack() << endl;

changeArrayBuku(2, "Bahasa Jerman");

cetakArrayBuku();
cout << "\n";

destroyArrayBuku();

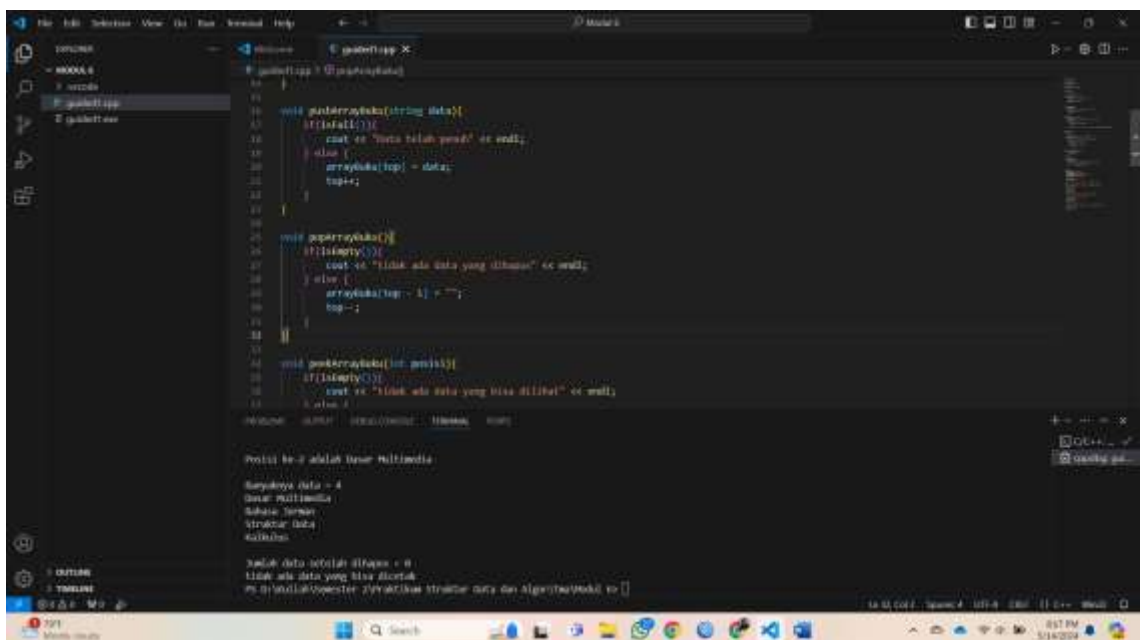
cout << "Jumlah data setelah dihapus = " << countStack() << endl;

cetakArrayBuku();

return 0;
}

```

Screenshoot Program:



Deskripsi Program:

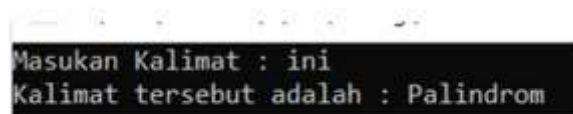
Program di atas adalah implementasi sederhana dari struktur data stack menggunakan array dalam bahasa C++. Program ini mendeklarasikan sebuah array ``arrayBuku`` berukuran lima untuk menyimpan data buku dengan indeks ``top`` sebagai penunjuk posisi teratas dari stack. Fungsi-fungsi yang ada meliputi ``isFull`` dan ``isEmpty`` untuk memeriksa apakah stack penuh atau kosong, ``pushArrayBuku`` untuk menambahkan data ke dalam stack, ``popArrayBuku`` untuk menghapus data dari stack, ``peekArrayBuku`` untuk melihat data pada posisi tertentu tanpa menghapusnya, ``countStack`` untuk menghitung jumlah elemen dalam stack, ``changeArrayBuku`` untuk mengubah data pada posisi tertentu, ``destroyArrayBuku`` untuk menghapus semua data dalam stack, dan ``cetakArrayBuku`` untuk mencetak seluruh isi stack. Program ini mengilustrasikan penggunaan stack dengan berbagai operasi dan mengecek kondisi penuh dan kosong dari stack.

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!

Kalimat : ini
Kalimat tersebut adalah polindrom

Kalimat : telkom
Kalimat tersebut adalah bukan polindrom



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

Program:

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

using namespace std;

string bersihkan_dan_ubah_ke_huruf_kecil_2031(string str) {
    string cleanStr;
    for (char &c : str) {
        if (isalpha(c)) {
            cleanStr += tolower(c);
        }
    }
    return cleanStr;
}

bool Palindrom_2031(string str) {
    stack<char> charStack;
    int length = str.length();

    for (int i = 0; i < length / 2; i++) {
        charStack.push(str[i]);
    }

    int start = length / 2;
    if (length % 2 != 0) {
        start++;
    }

    for (int i = start; i < length; i++) {
        if (charStack.empty() || str[i] != charStack.top()) {
            return false;
        }
    }
}
```

```

        charStack.pop();
    }

    return true;
}

int main() {
    char lanjutkan_2031;
    do {
        string input;
        cout << "\nMasukkan kata atau kalimat: ";
        getline(cin, input);

        string membersihkan_input_2031 =
bersihkan_dan_ubah_ke_huruf_kecil_2031(input);

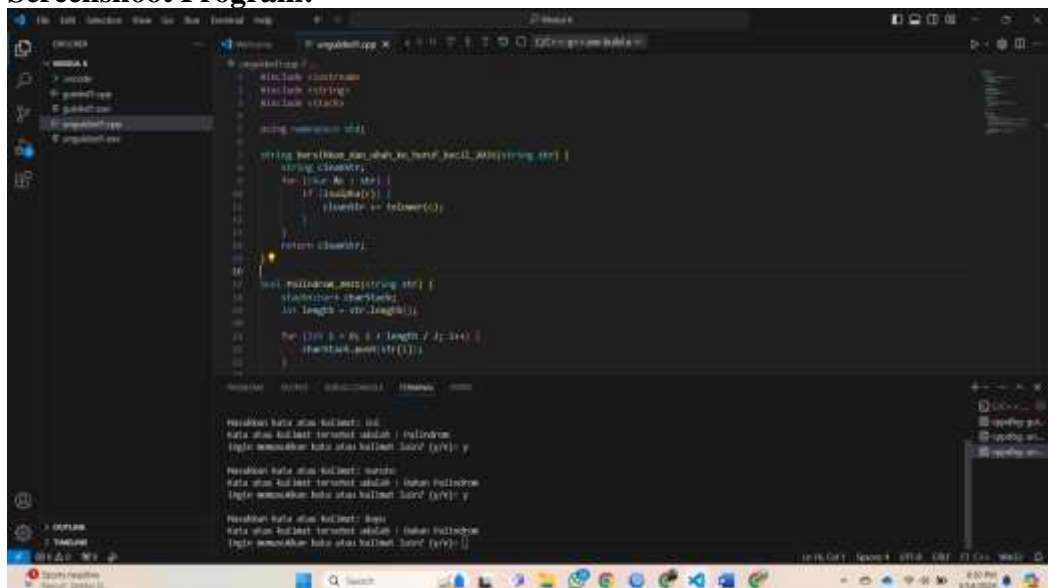
        if (Palindrom_2031(membersihkan_input_2031)) {
            cout << "Kata atau kalimat tersebut adalah : Palindrom" <<
endl;
        } else {
            cout << "Kata atau kalimat tersebut adalah : Bukan Palindrom"
<< endl;
        }

        cout << "Ingin memasukkan kata atau kalimat lain? (y/n): ";
        cin >> lanjutkan_2031;
        cin.ignore();
    } while (lanjutkan_2031 == 'y' || lanjutkan_2031 == 'Y');

    return 0;
}

```

Screenshoot Program:



Deskripsi Program:

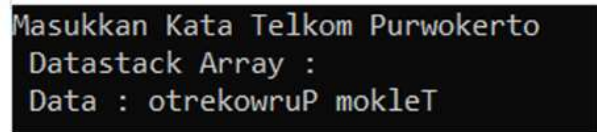
Program di atas adalah implementasi dalam bahasa C++ yang memeriksa apakah suatu kata atau kalimat adalah palindrom. Program ini meminta input dari pengguna, lalu membersihkan input tersebut dengan menghapus karakter non-alfabet dan mengubah semua huruf menjadi huruf kecil menggunakan fungsi ``bersihkan_dan_ubah_ke_huruf_kecil_2031``. Setelah itu, fungsi ``Palindrom_2031`` digunakan untuk menentukan apakah string yang telah dibersihkan adalah palindrom dengan menggunakan struktur data stack. Program akan mendorong karakter-karakter dari setengah bagian pertama string ke dalam stack, kemudian memeriksa karakter-karakter pada setengah bagian kedua dengan membandingkannya terhadap karakter yang ada di stack. Program ini akan terus meminta input dari pengguna sampai pengguna memilih untuk berhenti.

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 <string>
#include <stack>
#include <algorithm>

using namespace std;

string bersihkan_Dan_Kecilkan_Hurufnya_2031(string str) {
    string cleanStr;
    for (char &c : str) {
        if (isalpha(c)) {
            cleanStr += tolower(c);
        }
    }
    return cleanStr;
}

string balik_Kalimat_2031(string kalimat) {
    stack<char> tumpukanKarakter;
    string kalimat_Terbalik_2031;

    kalimat = bersihkan_Dan_Kecilkan_Hurufnya_2031(kalimat);

    for (char c : kalimat) {
        if (c != ' ') {
            tumpukanKarakter.push(c);
        } else {
            while (!tumpukanKarakter.empty()) {
                kalimat_Terbalik_2031 += tumpukanKarakter.top();
                tumpukanKarakter.pop();
            }
            kalimat_Terbalik_2031 += ' ';
        }
    }
}
```

```

    }

    while (!tumpukanKarakter.empty()) {
        kalimat_Terbalik_2031 += tumpukanKarakter.top();
        tumpukanKarakter.pop();
    }

    return kalimat_Terbalik_2031;
}

int main() {
    string kalimat_2031;
    char lanjutkan_2031;

    do {
        cout << "\nMasukkan kata atau kalimat (minimal 3 kata): ";
        getline(cin, kalimat_2031);

        int jumlah_Spasi_2031 = count(kalimat_2031.begin(),
kalimat_2031.end(), ' ');
        int jumlah_Kata_2031 = jumlah_Spasi_2031 + 1;

        if (jumlah_Kata_2031 < 3) {
            cout << "Masukkan minimal 3 kata!" << endl;
        } else {
            string kalimatTerbalik = balik_Kalimat_2031(kalimat_2031);
            cout << "Kalimat Terbalik: " << kalimatTerbalik << endl;
        }

        cout << "Apakah ingin memasukkan kata atau kalimat lagi? (y/n):
";
        cin >> lanjutkan_2031;
        cin.ignore();
    } while (lanjutkan_2031 == 'y' || lanjutkan_2031 == 'Y');

    return 0;
}

```

Program di atas adalah implementasi dalam bahasa C++ yang membalikkan setiap kata dalam sebuah kalimat yang diinput oleh pengguna, dengan syarat kalimat tersebut harus terdiri dari minimal tiga kata. Program ini pertama-tama meminta pengguna untuk memasukkan kalimat, kemudian menghitung jumlah kata dalam kalimat tersebut. Jika jumlah kata kurang dari tiga, program akan meminta pengguna untuk memasukkan kalimat yang valid. Jika valid, fungsi ``bersihkan_Dan_Kecilkan_Hurufnya_2031`` akan membersihkan kalimat dari karakter non-alfabet dan mengubahnya menjadi huruf kecil. Fungsi ``balik_Kalimat_2031`` menggunakan stack untuk membalikkan setiap kata dalam kalimat yang telah dibersihkan, dengan mendorong setiap karakter ke dalam stack hingga menemui spasi, lalu mempop karakter dari stack untuk mendapatkan kata terbalik. Hasil kalimat terbalik kemudian ditampilkan. Program akan terus meminta input dari pengguna sampai pengguna memilih untuk berhenti.

BAB IV

KESIMPULAN

Stack adalah struktur data yang bekerja berdasarkan prinsip LIFO (Last In, First Out), mirip dengan tumpukan piring di kafetaria di mana piring terakhir yang ditumpuk adalah piring pertama yang diambil. Dalam pemrograman, stack sering diimplementasikan menggunakan array atau linked list dan digunakan untuk berbagai keperluan seperti manajemen memori, rekursi, dan pemrosesan ekspresi matematika. Operasi utama pada stack meliputi push (menambahkan elemen), pop (menghapus elemen teratas), dan peek (melihat elemen teratas tanpa menghapusnya), yang semuanya membantu mengelola data secara terstruktur dan efisien.

Operasi pada stack tidak hanya terbatas pada push dan pop, tetapi juga mencakup fungsi seperti isEmpty (memeriksa apakah stack kosong), isFull (memeriksa apakah stack penuh), countStack (menghitung jumlah elemen dalam stack), dan destroyArrayBuku (mengosongkan stack). Dalam contoh program yang diberikan, stack digunakan untuk mengelola koleksi buku, dengan berbagai operasi yang memungkinkan penambahan, penghapusan, pengubahan, dan pemeriksaan elemen dalam stack. Aplikasi praktis dari stack sangat beragam, termasuk penyelesaian masalah palindrom dan pembalikan kata dalam kalimat.

Kelebihan stack termasuk kemampuannya mengelola data dengan metode LIFO, otomatisasi penghapusan objek yang tidak diperlukan, dan kontrol memori yang efisien. Namun, stack memiliki keterbatasan seperti kapasitas memori yang terbatas, kemungkinan terjadinya overflow saat menambahkan terlalu banyak elemen, dan ketidakmampuan untuk mengakses data secara acak. Stack ideal untuk situasi di mana urutan elemen dan akses ke elemen terbaru sangat penting, tetapi kurang cocok untuk skenario yang membutuhkan akses acak ke data.

Implementasi stack dalam pemrograman dapat dilihat pada dua contoh program yang dibahas. Program pertama mengelola koleksi buku menggunakan array sebagai stack, sementara program kedua memeriksa apakah suatu kalimat adalah palindrom dan membalikkan kata dalam kalimat menggunakan stack. Program-program ini menunjukkan bagaimana stack dapat diterapkan dalam situasi praktis untuk mengelola dan memproses data dengan efisien. Dengan memahami konsep dasar, operasi, dan aplikasi stack, serta kelebihan dan kekurangannya, programmer dapat memilih dan menerapkan struktur data ini dengan lebih efektif sesuai kebutuhan aplikasi mereka.

DAFTAR PUSTAKA

Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications

Modul 3 Stack diakses pada Tanggal 15 Mei 2024 dari

<https://sisfo.itp.ac.id/bahanajar/BahanAjar/Anisya/Modul%203%20-%20Tumpukan.pdf>

Modul 6 Stack Praktikum Struktur Data dan Algoritma Pemrograman

Struktur Data diakses dari <https://github.com/yunusfebriansyah/struktur-data/tree/main/Circular%20Single%20Linked%20List>

