LAPORAN PRAKTIKUM

MODUL VI
STACK

Disusun oleh:
Anita Nurazizah Agussalim
NIM: 2311102017

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

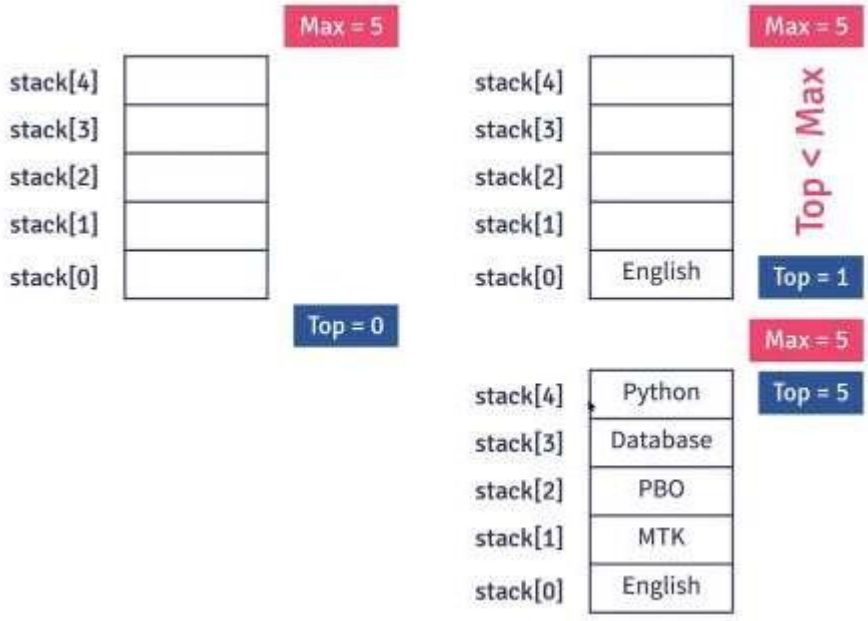
1. Mampu memahami konsep stack pada struktur data dan algoritma
2. Mampu mengimplementasikan operasi-operasi pada stack
3. Mampu memecahkan permasalahan dengan solusi stack

BAB II

DASAR TEORI

Stack adalah struktur data sederhana yang digunakan untuk menyimpan data (mirip dengan Linked Lists). Dalam tumpukan, urutan kedatangan data penting. Sebuah tumpukan piring di kafetaria adalah contoh bagus dari tumpukan. Piring ditambahkan ke tumpukan saat mereka dibersihkan dan ditempatkan di bagian atas. Ketika sebuah piring dibutuhkan, diambil dari bagian atas tumpukan. Piring pertama yang ditempatkan di tumpukan adalah yang terakhir digunakan.

Definisi: Sebuah tumpukan adalah daftar terurut di mana penyisipan dan penghapusan dilakukan di satu ujung, disebut atas. Elemen terakhir yang dimasukkan adalah yang pertama dihapus. Oleh karena itu, disebut daftar Last in First out (LIFO).



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

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

b. Pop (Keluarkan): Menghapus elemen dari posisi paling atas atau ujung tumpukan.
c. Top (Atas): Mendapatkan nilai atau melihat elemen teratas pada tumpukan tanpamenghapusnya.
d. IsEmpty (Kosong): Memeriksa apakah tumpukan kosong atau tidak.
e. IsFull (Penuh): Memeriksa apakah tumpukan penuh atau tidak (terutama padaimplementasi 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 tanpamenghapusnya.
h. Clear (Hapus Semua): Mengosongkan atau menghapus semua elemen daritumpukan.
i. Search (Cari): Mencari keberadaan elemen tertentu dalam tumpukan.

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;
}
```

Screenshot Output

```
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
```

Deskripsi program

Program di atas adalah implementasi dari struktur data stack menggunakan array untuk menyimpan koleksi buku. Stack memiliki kapasitas maksimal lima buku dan menyediakan beberapa operasi dasar: ‘pushArrayBuku’ untuk menambahkan buku ke stack, ‘popArrayBuku’ untuk menghapus buku dari stack, ‘peekArrayBuku’ untuk melihat buku pada posisi tertentu, ‘countStack’ untuk menghitung jumlah buku dalam stack, ‘changeArrayBuku’ untuk mengganti buku pada posisi tertentu, ‘destroyArraybuku’ untuk menghapus semua buku dalam stack, dan ‘cetakArrayBuku’ untuk mencetak semua buku dalam stack. Program ini juga menyediakan fungsi untuk memeriksa apakah stack penuh (‘isFull’) atau kosong (‘isEmpty’). Contoh penggunaan fungsi-fungsi tersebut dilakukan dalam fungsi 'main()’.

LATIHAN KELAS - UNGUIDED

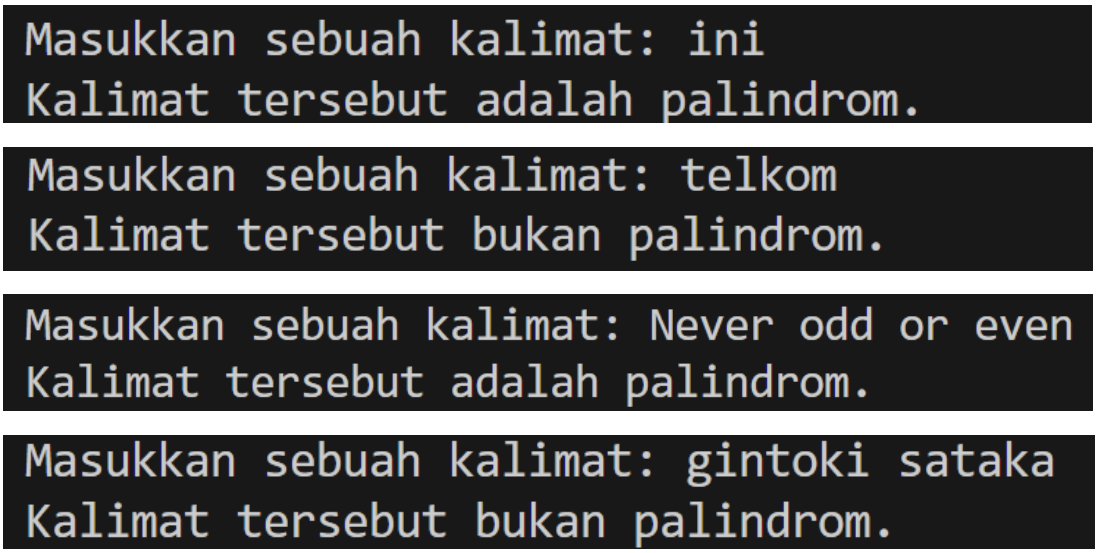
1. Buatlah program untuk menentukan apakah kalimat tersebut yang diinputkan dalam program stack adalah palindrom/tidak. Palindrom kalimat yang dibaca dari depan dan belakang sama. Jelaskan bagaimana cara kerja programnya.

Source code

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

using namespace std;

bool isPalindrome(string str) {
    string cleanedStr = "";
    for (char c : str) {
        if (isalnum(c)) {
            cleanedStr += tolower(c);
        }
    }

    stack<char> charStack;

    for (char c : cleanedStr) {
        charStack.push(c);
    }

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

```
            }

            return true;
        }

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

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

            return 0;
}
```

OUTPUT PROGRAM



CARA KERJA PROGRAM

Misalkan pengguna memasukkan kalimat “Never odd or even”:

1. Kalimat yang mengandung karakter non-alfanumerik akan diubah semua huruf menjadi huruf kecil. Ini memastikan bahwa perbandingan dilakukan hanya pada huruf dan angka, serta mengabaikan perbedaan huruf besar dan kecil. Pada kasus ini, kalimat dibersihkan menjad “neveroddoreven”.
2. Program memasukkan setiap karakter dari “neveroddoreven” yang telah dibersihkan ke dalam stack. Stack mengikuti prinsip LIFO (Last In, First Out), yang berarti karakter terakhir yang dimasukkan akan menjadi yang pertama diambil.
3. Program membandingkan setiap karakter dari “neveroddoreven” yang dibersihkan dengan karakter yang diambil dari stack. Karena stack mengikuti prinsip LIFO, ini memungkinkan kita membandingkan string tersebut dengan cara yang terbalik.
4. Karena semua karakter cocok, sehingga program mengembalikan bahwa kalimat tersebut adalah palindrom.

DESKRIPSI PROGRAM

Program di atas menentukan apakah sebuah kalimat adalah palindrom menggunakan stack. Program meminta pengguna memasukkan kalimat, lalu membersihkan kalimat tersebut dengan menghilangkan karakter non-alfanumerik dan mengubah semua huruf menjadi huruf kecil. Setelah itu, setiap karakter dari kalimat yang telah dibersihkan dimasukkan ke dalam stack. Program kemudian membandingkan karakter dari awal kalimat dengan karakter yang diambil dari stack secara terbalik. Jika semua karakter cocok, kalimat tersebut adalah palindrom; jika tidak, bukan palindrom. Hasilnya ditampilkan kepada pengguna.

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 <sstream>

using namespace std;

string reverseString(string str) {
    stack<char> charStack;
    for (char c : str) {
        charStack.push(c);
    }

    string reversedStr = "";
    while (!charStack.empty()) {
        reversedStr += charStack.top();
        charStack.pop();
    }
    return reversedStr;
}

void reverseSentence(string str) {
    stringstream ss(str);
    string word;
    string reversedSentence = "";
    int wordCount = 0;

    while (ss >> word) {
        wordCount++;
    }
```

```
    ss.clear();
    ss.str(str);

    if (wordCount < 3) {
        cout << "Kalimat harus memiliki minimal 3 kata." << endl;
        return;
    }

    while (ss >> word) {
        string reversedWord = reverseString(word);
        reversedSentence = reversedWord + " " + reversedSentence;
    }

    if (!reversedSentence.empty() &&
reversedSentence[reversedSentence.size() - 1] == ' ') {
        reversedSentence.pop_back();
    }

    cout << "Kalimat: " << str << endl;
    cout << "Hasil: " << reversedSentence << endl;
}

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

    reverseSentence(input);

    return 0;
}
```

OUTPUT PROGRAM

```
Masukkan sebuah kalimat dengan minimal 3 kata: IT Telkom Purwokerto
Kalimat: IT Telkom Purwokerto
Hasil: otrekowruP mokleT TI
```

```
Masukkan sebuah kalimat dengan minimal 3 kata: Saya suka nasi goreng
Kalimat: Saya suka nasi goreng
Hasil: gnerog isan akus ayaS
```

```
Masukkan sebuah kalimat dengan minimal 3 kata: Punya Anita
Kalimat harus memiliki minimal 3 kata.
```

PENJELASAN SOURCE CODE DAN OUTPUT PROGRAM

1. Fungsi ‘reverseString’
    - Fungsi ini menerima sebuah string (‘str’) dan mengembalikan string yang urutan karakternya dibalik.
    - Setiap karakter dalam string dimasukkan ke dalam stack.
    - Karakter kemudian diambil dari stack satu per satu, sehingga urutan karakter menjadi terbalik.

```
string reverseString(string str) {
    stack<char> charStack;
    for (char c : str) {
        charStack.push(c);
    }

    string reversedStr = "";
    while (!charStack.empty()) {
        reversedStr += charStack.top();
        charStack.pop();
    }
    return reversedStr;
}
```

2. Fungsi ‘reverseSentence’:
    - Fungsi ini menerima sebuah kalimat (str) dan membalik urutan kata dalam kalimat serta urutan huruf dalam setiap kata.
    - Kalimat diproses menggunakan stringstream untuk memisahkan kata-kata dan menghitung jumlah kata.
    - Jika jumlah kata kurang dari tiga, program menampilkan pesan kesalahan dan tidak melanjutkan proses.
    - Jika jumlah kata cukup, setiap kata dibalik menggunakan fungsi reverseString dan urutan kata dibalik.
    - Hasil akhir ditampilkan dengan format yang diinginkan.

```
void reverseSentence(string str) {
    stringstream ss(str);
    string word;
    string reversedSentence = "";
    int wordCount = 0;

    // Hitung jumlah kata dalam kalimat
    while (ss >> word) {
        wordCount++;
    }

    // Reset stringstream untuk digunakan kembali
    ss.clear();
    ss.str(str);

    // Memastikan kalimat memiliki minimal 3 kata
    if (wordCount < 3) {
        cout << "Kalimat harus memiliki minimal 3 kata." << endl;
        return;
    }
```

```
    // Membalik urutan huruf dalam setiap kata dan urutan kata
dalam kalimat
    while (ss >> word) {
        string reversedWord = reverseString(word);
        reversedSentence = reversedWord + " " + reversedSentence;
    }

    // Menghapus spasi di akhir kalimat
    if (!reversedSentence.empty() &&
reversedSentence[reversedSentence.size() - 1] == ' ') {
        reversedSentence.pop_back();
    }

    cout << "Kalimat: " << str << endl;
    cout << "Hasil: " << reversedSentence << endl;
}
```

3. Fungsi main:
    - Fungsi ini meminta pengguna untuk memasukkan sebuah kalimat dengan minimal tiga kata.
    - Kalimat yang dimasukkan kemudian diproses oleh fungsi reverseSentence.

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

    reverseSentence(input);

    return 0;
}
```

4. Output

   a). Jika kalimat memiliki kurang dari tiga kata:

```
Kalimat harus memiliki minimal 3 kata.
```

   b). Jika kalimat memiliki tiga kata atau lebih, misalnya input: "Saya suka nasi goreng":

```
Kalimat: Saya suka nasi goreng
Hasil: gnerog isan akus ayaS
```

   Program ini memastikan bahwa kalimat terdiri dari minimal tiga kata sebelum memproses pembalikan kata dan huruf. Fungsi reverseString digunakan untuk membalik huruf dalam setiap kata, dan reverseSentence membalik urutan kata serta huruf dalam kalimat tersebut.

DESKRIPSI PROGRAM

Program di atas adalah program yang membalik urutan kata dan huruf dalam sebuah kalimat. Program ini meminta pengguna untuk memasukkan sebuah kalimat dengan minimal tiga kata. Jika kalimat memiliki kurang dari tiga kata, program akan menampilkan pesan kesalahan. Kemudian, program akan memproses kalimat tersebut dengan membalik urutan huruf dalam setiap kata menggunakan fungsi ‘reverseString’ yang mengimplementasikan stack. Setelah itu, urutan kata dalam kalimat juga dibalik menggunakan stack. Hasil akhirnya ditampilkan dalam format yang diinginkan. Program ini memanfaatkan struktur data stack dan stringstream untuk memisahkan kalimat menjadi kata-kata serta menghitung jumlah kata dalam kalimat.

BAB IV

KESIMPULAN

Stack adalah struktur data sederhana yang mengikuti prinsip Last In First Out (LIFO), di mana elemen terakhir yang dimasukkan adalah yang pertama dihapus. Operasi dasar pada stack meliputi push (menambahkan elemen), pop (menghapus elemen), top (melihat elemen teratas), isEmpty (memeriksa apakah tumpukan kosong), isFull (memeriksa apakah tumpukan penuh), size (mengembalikan jumlah elemen), peek (melihat nilai elemen tanpa menghapusnya), clear (menghapus semua elemen), dan search (mencari keberadaan elemen tertentu). Analogi tumpukan piring di kafetaria memberikan gambaran yang baik tentang konsep ini, di mana piring yang ditambahkan terakhir adalah yang pertama diambil. Konsep ini penting dalam pemrograman untuk menyimpan dan mengakses data dalam urutan tertentu, dan operasi-operasi tersebut membentuk dasar implementasi stack dalam berbagai aplikasi.

DAFTAR PUSTAKA

Asisten Praktikum. (2024). MODUL 6: STACK.

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