LAPORAN PRAKTIKUM

MODUL

STACK

Nama :

Fajar Budiawan (2311104039)

Dosen :

Yudha Islami Sulistya S.Kom, M.Cs

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

TUGAS PENDAHULUAN

```h
#ifndef STACK_H
#define STACK_H

#include <iostream>
using namespace std;

typedef char infotype;

struct Stack {
    infotype info[15];
    int Top;
};

// Fungsi dan Prosedur untuk ADT Stack
void createStack(Stack &S);
bool isEmpty(Stack S);
bool isFull(Stack S);
void push(Stack &S, infotype x);
infotype pop(Stack &S);
void printInfo(Stack S, int num_to_pop);

#endif
```

File header `stack.h` mendefinisikan struktur data stack dalam C++. Terdapat pengaturan guard clauses untuk mencegah duplikasi include, serta menyertakan library `iostream` untuk fungsi input/output. Tipe data `infotype` didefinisikan sebagai `char`, dan struktur `Stack` berisi array `info[15]` untuk menyimpan elemen dan variabel `Top` untuk menunjukkan indeks elemen teratas. Fungsi-fungsi yang dideklarasikan meliputi `createStack` untuk menginisialisasi stack, `isEmpty` dan `isFull` untuk memeriksa keadaan stack, `push` untuk menambahkan elemen, `pop` untuk menghapus elemen teratas, dan `printInfo` untuk mencetak informasi dari stack. Dengan demikian, `stack.h` menyediakan antarmuka yang jelas untuk operasi dasar pada stack.

```cpp
#include "stack.h"

// Membuat stack baru dengan Top = 0
void createStack(Stack &S) {
    S.Top = 0;
}

// Mengecek apakah stack kosong
bool isEmpty(Stack S) {
    return (S.Top == 0);
}

// Mengecek apakah stack penuh
bool isFull(Stack S) {
    return (S.Top == 15);
}

// Menambahkan elemen ke dalam stack
void push(Stack &S, infotype x) {
    if (!isFull(S)) {
        S.Top++;
        S.info[S.Top] = x;
    } else {
        cout << "Stack penuh!" << endl;
    }
}

// Mengeluarkan elemen dari stack
infotype pop(Stack &S) {
    if (!isEmpty(S)) {
        infotype x = S.info[S.Top];
        S.Top--;
        return x;
    } else {
        cout << "Stack kosong!" << endl;
        return '\0';  // Mengembalikan karakter kosong jika stack kosong
    }
}

// Menampilkan semua elemen di stack dari bawah ke atas dan melakukan pop sesuai jumlah tertentu
void printInfo(Stack S, int num_to_pop) {
    // Cetak isi stack awal
    cout << "Isi stack awal: ";
    for (int i = 1; i <= S.Top; i++) {
        cout << S.info[i] << " ";
    }
    cout << endl;

    // Pop sejumlah elemen sesuai permintaan dan tampilkan sisanya
    cout << "Isi stack setelah pop: ";
    for (int i = 0; i < num_to_pop; i++) {
        pop(S);
    }
    for (int i = 1; i <= S.Top; i++) {
        cout << S.info[i] << " ";
    }
    cout << endl;
}
```

Kode di atas mengimplementasikan fungsi dasar untuk struktur data stack dalam C++. Fungsi `createStack` menginisialisasi stack dengan `Top` bernilai 0. Fungsi `isEmpty` dan `isFull` memeriksa keadaan stack, sedangkan `push` menambahkan elemen jika stack tidak penuh. Fungsi `pop` menghapus elemen teratas jika stack tidak kosong. Fungsi `printInfo` menampilkan isi stack, melakukan pop sesuai jumlah yang diminta, dan mencetak sisa elemen. Secara keseluruhan, kode ini menyediakan fungsionalitas dasar untuk pengelolaan stack.

```cpp
#include <iostream>
#include <string>
#include "stack.h"

// Fungsi untuk menguji stack berdasarkan sisa digit terakhir NIM MOD 4
void testStack(int remainder) {
    Stack S;
    createStack(S);

    string data;
    int num_to_pop;

    // Menentukan data dan jumlah elemen yang akan di-pop berdasarkan remainder
    if (remainder == 0) {
        data = "IFLABJAYA";
        num_to_pop = 5;
    } else if (remainder == 1) {
        data = "HALOBANDUNG";
        num_to_pop = 7;
    } else if (remainder == 2) {
        data = "PERCAYADIRI";
        num_to_pop = 8;
    } else if (remainder == 3) {
        data = "STRUKTURDATA";
        num_to_pop = 9;
    }

    // Menyimpan data ke dalam stack
    for (char c : data) {
        push(S, c);
    }

    // Cetak hasil sesuai dengan jumlah elemen yang di-pop
    printInfo(S, num_to_pop);
}

int main() {
    int NIM_last_digit = 2;
    int remainder = NIM_last_digit % 4;

    testStack(remainder);

    return 0;
}
```

Kode ini menguji struktur data stack berdasarkan sisa digit terakhir NIM MOD 4. Fungsi `testStack` menginisialisasi stack dan menentukan string `data` serta jumlah elemen yang akan di-pop sesuai dengan nilai `remainder`. Karakter dari string disimpan dalam stack menggunakan `push`, kemudian isi stack dicetak setelah melakukan pop dengan fungsi `printInfo`. Di dalam `main`, nilai `NIM_last_digit` dihitung untuk menentukan `remainder` dan memanggil `testStack`. Kode ini menunjukkan penggunaan stack untuk menyimpan dan memanipulasi karakter.

Output

```
PS D:\STRUKTUR DATA\07_Stack> cd "d:\STRUKTUR DATA\07_Stack\TP\" ; if ($?)
Digit terakhir NIM MOD 4 sisa 0 :
Output:
I F L A B J A Y A
I F L A
PS D:\STRUKTUR DATA\07_Stack\TP>
```