LAPORAN PRAKTIKUM

PERTEMUAN 7

STACK

Nama :

Resita Istania Purwanto (2311104037)

Dosen :

Yudha Islami Sulistya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TP

Kode stack.h

```h
#ifndef STACK_H
#define STACK_H

typedef char infotype;

Codeium: Refactor | Explain
struct stack {
    infotype info[15]; // Array untuk menyimpan elemen stack
    int Top; // Indeks elemen paling atas
};

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);

#endif
```

stack.cpp

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

Codeium: Refactor | Explain | Generate Function Comment | ×
void createStack(stack &S) {
    S.Top = 0; // Inisialisasi stack kosong
}

Codeium: Refactor | Explain | Generate Function Comment | ×
bool isEmpty(stack S) {
    return S.Top == 0; // Memeriksa apakah stack kosong
}

Codeium: Refactor | Explain | Generate Function Comment | ×
bool isFull(stack S) {
    return S.Top == 15; // Memeriksa apakah stack penuh
}

Codeium: Refactor | Explain | Generate Function Comment | ×
void push(stack &S, infotype x) {
    if (!isFull(S)) {
        S.info[S.Top] = x; // Masukkan elemen pada posisi Top
        S.Top++; // Naikkan Top
    } else {
        std::cout << "Stack penuh!" << std::endl; // Penanganan stack penuh
    }
}

Codeium: Refactor | Explain | Generate Function Comment | ×
infotype pop(stack &S) {
    if (!isEmpty(S)) {
        S.Top--; // Turunkan Top
        return S.info[S.Top]; // Kembalikan elemen yang di-pop
    }
    std::cout << "Stack kosong!" << std::endl; // Penanganan stack kosong
    return '\0'; // Kembali karakter null jika stack kosong
}

Codeium: Refactor | Explain | Generate Function Comment | ×
void printInfo(stack S) {
    for (int i = S.Top - 1; i >= 0; i--) {
        std::cout << S.info[i] << " "; // Cetak dari Top ke 0
    }
    std::cout << std::endl;
}
```

main.cpp

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

Codeium: Refactor | Explain | Generate Function Comment | ×
void testStack(char data[], int size) {
    stack S;
    createStack(S); // Membuat stack baru

    for (int i = 0; i < size; i++) {
        push(S, data[i]); // Menambahkan elemen ke stack
    }

    std::cout << "Isi stack awal: ";
    printInfo(S); // Menampilkan isi stack awal

    // Pop beberapa elemen (disesuaikan dengan data)
    for (int i = 0; i < size / 2; i++) { // Misal kita pop setengah dari ukuran
        pop(S);
    }

    std::cout << "Isi stack sesudah pop: ";
    printInfo(S); // Menampilkan isi stack setelah pop
}

Codeium: Refactor | Explain | Generate Function Comment | ×
int main() {
    // Digit terakhir NIM MOD 4 sisa 0
    char data0[] = {'A', 'Y', 'A', 'J', 'B', 'A', 'L', 'F', 'I'}; // Ubah sesuai kebutuhan
    std::cout << "Digit terakhir NIM MOD 4 sisa 0:" << std::endl;
    testStack(data0, 9); // Menjalankan pengujian untuk data0

    // Digit terakhir NIM MOD 4 sisa 1
    char data1[] = {'G', 'N', 'U', 'D', 'N', 'A', 'B', 'O', 'L', 'A', 'H'};
    std::cout << "Digit terakhir NIM MOD 4 sisa 1:" << std::endl;
    testStack(data1, 11); // Menjalankan pengujian untuk data1

    // Digit terakhir NIM MOD 4 sisa 2
    char data2[] = {'I', 'R', 'I', 'D', 'A', 'Y', 'A', 'C', 'R', 'E', 'P'};
    std::cout << "Digit terakhir NIM MOD 4 sisa 2:" << std::endl;
    testStack(data2, 11); // Menjalankan pengujian untuk data2

    // Digit terakhir NIM MOD 4 sisa 3
    char data3[] = {'A', 'T', 'A', 'D', 'R', 'U', 'T', 'K', 'U', 'R', 'T', 'S'};
    std::cout << "Digit terakhir NIM MOD 4 sisa 3:" << std::endl;
    testStack(data3, 12); // Menjalankan pengujian untuk data3

    return 0;
}
```

Output:

```
Digit terakhir NIM MOD 4 sisa 0:
Isi stack awal: I F L A B J A Y A
Isi stack sesudah pop: B J A Y A
Digit terakhir NIM MOD 4 sisa 1:
Isi stack awal: H A L O B A N D U N G
Isi stack sesudah pop: A N D U N G
Digit terakhir NIM MOD 4 sisa 2:
Isi stack awal: P E R C A Y A D I R I
Isi stack sesudah pop: Y A D I R I
Digit terakhir NIM MOD 4 sisa 3:
Isi stack awal: S T R U K T U R D A T A
Isi stack sesudah pop: U R D A T A
```

Penjelasan:

- Struktur stack: Menyimpan elemen dalam array info dan indeks Top.
- Fungsi Utama:

a. createStack: Inisialisasi stack kosong.

b. isEmpty: Cek jika stack kosong.
c. isFull: Cek jika stack penuh.
d. push: Tambah elemen ke stack.
e. pop: Hapus dan kembalikan elemen teratas.
f. printInfo: Cetak semua elemen dari top ke bottom.
    - Fungsi testStack
        a. Menguji stack dengan memasukkan elemen dari array, mencetak isi awal, lalu menghapus setengah elemen dan mencetak isi setelah pop.
            - Fungsi main
                a. Mempersiapkan data untuk berbagai kondisi (sisa 0 hingga 3) dan memanggil testStack untuk masing-masing dataset.

- Output : Ketika dijalankan, kode mencetak isi stack sebelum dan setelah elemen dihapus, menunjukkan cara kerja stack**.**

II. UNGUIDED

Latihan modul

1. Palindrom

Kode:

```cpp
#include <iostream>
#include <string>

using namespace std;

// Fungsi untuk memeriksa apakah kata adalah palindrom
Codeium: Refactor | Explain | ×
bool isPalindrome(const string &word) {
    int n = word.length();
    for (int i = 0; i < n / 2; i++) {
        if (word[i] != word[n - 1 - i]) {
            return false; // Jika ada karakter yang berbeda, bukan palindrom
        }
    }
    return true; // Jika semua karakter sama, adalah palindrom
}

Codeium: Refactor | Explain | Generate Function Comment | ×
int main() {
    string word;
    cout << "Masukkan kata: ";
    cin >> word; // Menerima input kata dari pengguna

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

    return 0;
}
```

Output:

```
Masukkan kata: level
Kata tersebut adalah palindrom.
```

Penjelasan: Program C++ ini memeriksa apakah sebuah kata adalah palindrom.

a. Library:
- #include <iostream>: Untuk input/output.
- #include <string>: Untuk penggunaan string.

b. Fungsi isPalindrome:
   - Menerima string word, membandingkan karakter dari awal dan akhir.
   - Mengembalikan true jika kata adalah palindrom, false jika tidak.

c. Fungsi main:
   - Menerima input kata dari pengguna.
   - Memanggil isPalindrome untuk memeriksa kata.
   - Menampilkan hasil pemeriksaan.
   - Program ini menentukan apakah kata dibaca sama dari depan dan belakang

2. Kode:

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

using namespace std;

// Fungsi untuk membalik kalimat
Codeium: Refactor | Explain | ×
void reverseSentence(const string &input) {
    stack<string> s; // Menggunakan stack untuk menyimpan kata
    stringstream ss(input); // Menggunakan stringstream untuk memisahkan kata
    string word;

    // Memasukkan setiap kata ke dalam stack
    while (ss >> word) {
        s.push(word);
    }

    // Menampilkan kalimat dengan kata-kata terbalik
    cout << "Kalimat setelah dibalik: ";
    while (!s.empty()) {
        cout << s.top() << " "; // Mengeluarkan kata teratas dari stack
        s.pop(); // Menghapus kata dari stack
    }
    cout << endl; // Menambahkan baris baru setelah kalimat terbalik
}

Codeium: Refactor | Explain | Generate Function Comment | ×
int main() {
    string input;
    cout << "Masukkan kalimat dengan minimal 3 kata: ";
    getline(cin, input); // Menerima input kalimat dari pengguna
```

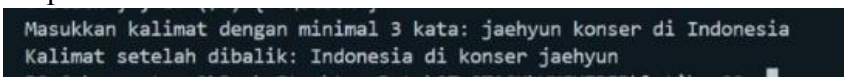
```cpp
    // Memastikan kalimat memiliki minimal 3 kata
    stringstream ss(input);
    string word;
    int wordCount = 0;

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

    if (wordCount < 3) {
        cout << "Harap masukkan kalimat dengan minimal 3 kata." << endl;
    } else {
        reverseSentence(input); // Memanggil fungsi untuk membalik kalimat
    }

    return 0;
}
```

Output:



Penjelasan:

- <iostream>: Untuk input/output.
- <stack>: Untuk struktur data stack.
- <sstream>: Untuk memisahkan kata.
- <string>: Untuk penggunaan string.
- reverseSentence: Memisahkan kalimat menjadi kata, menyimpan ke stack, dan membalik urutannya.
- main: Menerima input, memeriksa jumlah kata (minimal 3), dan memanggil reverseSentence.
- Program ini menggunakan stack untuk membalik urutan kata dalam kalimat.