LAPORAN PRAKTIKUM

PERTEMUAN 7

Nama :

Razhendriya Vania Ramadhan Suganjarsarwat (2311104048)

Dosen :

WAHYU ANDI SAPUTRA

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

Memahami konsep stack, menerapkan operasi dasar stack, dan menyelesaikan masalah yang relevan menggunakan stack.

II. DASAR TEORI

Stack adalah struktur data berbasis prinsip Last-In, First-Out (LIFO), di mana elemen terakhir yang dimasukkan adalah elemen pertama yang dikeluarkan. Stack digunakan dalam berbagai aplikasi, seperti manajemen memori, evaluasi ekspresi aritmatika, dan panggilan fungsi dalam pemrograman.

Operasi dasar pada stack meliputi:

- **Push: Menambahkan elemen baru di atas stack.**
- **Pop: Menghapus elemen teratas.**
- **Peek: Melihat elemen teratas tanpa menghapusnya.**
- **IsEmpty: Mengecek apakah stack kosong.**
- **IsFull: Mengecek apakah stack penuh.**

Stack dapat diimplementasikan menggunakan array atau linked list. Implementasi menggunakan array memiliki kapasitas tetap, sedangkan linked list lebih fleksibel karena tidak memerlukan batasan kapasitas.

III. GUIDED

Penjelasan Kode

1. **Deklarasi Stack:**
   - **Kelas Stack memiliki dua atribut privat:**
     - **int top: Penunjuk ke elemen terakhir dalam stack.**
     - **int arr[MAX]: Array untuk menyimpan elemen stack.**
   - **Ukuran maksimum stack didefinisikan dengan #define MAX 100.**
2. **Konstruktor:**
   - **Inisialisasi top ke -1, yang berarti stack kosong.**
3. **Fungsi isFull():**
   - **Memeriksa apakah stack penuh. Jika top mencapai indeks terakhir (MAX - 1), maka stack penuh.**
4. **Fungsi isEmpty():**
   - **Memeriksa apakah stack kosong. Jika top == -1, berarti tidak ada elemen dalam stack.**
5. **Fungsi push(int value):**
   - **Menambahkan elemen ke stack.**
   - **Jika stack penuh, cetak pesan kesalahan.**
   - **Jika tidak, tambahkan elemen di posisi berikutnya (arr[++top]).**
6. **Fungsi pop():**
   - **Menghapus elemen dari stack.**
   - **Jika stack kosong, cetak pesan kesalahan.**
   - **Jika tidak, kurangi top.**
7. **Fungsi peek():**
   - **Mengembalikan elemen di puncak stack.**
   - **Jika stack kosong, cetak pesan dan kembalikan -1 sebagai nilai sentinel.**
8. **Fungsi display():**
   - **Menampilkan semua elemen dalam stack dari atas ke bawah.**
   - **Jika stack kosong, cetak pesan.**
9. **Fungsi main():**
   - **Membuat objek stack (Stack s).**
   - **Melakukan operasi seperti push, pop, peek, dan display untuk menunjukkan cara kerja stack.**

```cpp
#include <iostream>
using namespace std;

// Kelas Node untuk merepresentasikan elemen stack
class Node {
public:
    int data;
    Node* next;
};

// Kelas Stack menggunakan linked list
class Stack {
private:
        // Pointer ke elemen teratas
    Node* top;stack
public:
    // Konstruktor
    Stack() {
        top = nullptr;
    }

    // Fungsi untuk memeriksa apakah stack kosong
    bool isEmpty() {
        return top == nullptr;
    }

    // Fungsi untuk menambahkan elemen ke stack
    void push(int x) {
        Node* newNode = new Node();
        newNode->data = x;
        newNode->next = top;
        top = newNode;
        cout << x << " pushed to stack.\n";
    }

    // Fungsi untuk menghapus elemen dari stack
    void pop() {
        if (isEmpty()) {
            cout << "Stack Underflow\n";
            return;
        }
        Node* temp = top;
        top = top->next;
        cout << temp->data << " popped from stack.\n";
        delete temp;
    }

    // Fungsi untuk melihat elemen teratas stack
    int peek() {
        if (isEmpty()) {
            cout << "Stack is empty\n";
            return -1; // Nilai sentinel jika stack kosong
        }
        return top->data;
    }

    // Fungsi untuk menampilkan elemen stack
    void display() {
        if (isEmpty()) {
            cout << "Stack is empty\n";
            return;
        }
        Node* current = top;
        cout << "Stack elements: ";
        while (current) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }
};

// Fungsi utama untuk menjalankan program
int main() {
    Stack s;

    // Menambahkan elemen ke stack
    s.push(10);
    s.push(20);
    s.push(30);

    // Menampilkan elemen dalam stack
    s.display();

    // Melihat elemen teratas
    cout << "Top element: " << s.peek() << endl;

    // Menghapus elemen dari stack
    s.pop();

    // Menampilkan elemen setelah pop
    s.display();

    return 0;
}
```

Penjelasan Kode

1. **Kelas Node:**
    - **Setiap elemen stack diwakili oleh sebuah Node.**
    - **Memiliki atribut:**
        - **data: Menyimpan nilai elemen stack.**
        - **next: Menunjuk ke elemen berikutnya dalam stack.**
2. **Kelas Stack:**
    - **Memiliki pointer top untuk menunjuk ke elemen teratas stack.**
    - **Metode:**
        - **isEmpty: Memeriksa apakah stack kosong.**
        - **push: Menambahkan elemen baru ke stack dengan membuat node baru dan menjadikannya elemen teratas.**
        - **pop: Menghapus elemen teratas dari stack.**
        - **peek: Mengembalikan nilai elemen teratas tanpa menghapusnya.**
        - **display: Menampilkan semua elemen dalam stack.**
3. **Fungsi main():**
    - **Membuat objek stack (Stack s).**
    - **Menjalankan berbagai operasi seperti:**
        - **Menambahkan elemen (push).**
        - **Menampilkan elemen stack (display).**
        - **Melihat elemen teratas (peek).**
        - **Menghapus elemen (pop).**

IV. UNGUIDED

```cpp
#include <iostream>
#include <stack>
using namespace std;

bool isPalindrome(string str) {
    stack<char> s;
    for (char c : str) {
        if (isalpha(c)) { // Abaikan karakter non-alfabet
            s.push(tolower(c));
             }
       }

    for (char c : str) {
        if (isalpha(c)) {
            if (s.top() != tolower(c)) {
                return false;
                   }
            s.pop();
             }
       }
    return true;
 }

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

    if (isPalindrome(input)) {
        cout << "Kalimat adalah palindrom.\n";
    } else {
        cout << "Kalimat bukan palindrom.\n";
       }
    return 0;
 }
```

Penjelasan Program:

- **Program menggunakan stack untuk menyimpan karakter alfabet dari input.**
- **Proses perbandingan dilakukan dengan membandingkan karakter teratas stack dengan karakter input secara berurutan.**
- **Hanya karakter alfabet yang diproses, dan perbandingan tidak memperhatikan huruf besar atau kecil.**

```cpp
#include <iostream>
#include <stack>
#include <sstream>
using namespace std;

void reverseWords(string sentence) {
    stack<string> s;
    stringstream ss(sentence);
    string word;

    while (ss >> word) {
        s.push(word);
      }

    while (!s.empty()) {
        cout << s.top() << " ";
        s.pop();
      }
    cout << endl;
}

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

    cout << "Kalimat setelah dibalik: ";
    reverseWords(input);

    return 0;
}
```

Penjelasan Program:

- **Program membaca input kalimat dan memecahnya menjadi kata-kata menggunakan stringstream.**
- **Kata-kata dimasukkan ke dalam stack, lalu dikeluarkan untuk mencetak kalimat dengan urutan terbalik.**

V. KESIMPULAN

Laporan ini membahas konsep dasar dan implementasi stack. Berdasarkan program yang dibuat:

- **Stack mempermudah implementasi algoritma seperti palindrom checker dan pembalik kata.**
- **Operasi dasar seperti push, pop, dan peek memungkinkan manipulasi data yang efisien dengan prinsip LIFO.**