LAPORAN PRAKTIKUM
Modul 7
STACK

Disusun Oleh:

Muhammad Shafiq Rasuna - 2311104043

Kelas :

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd, M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. Tujuan

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

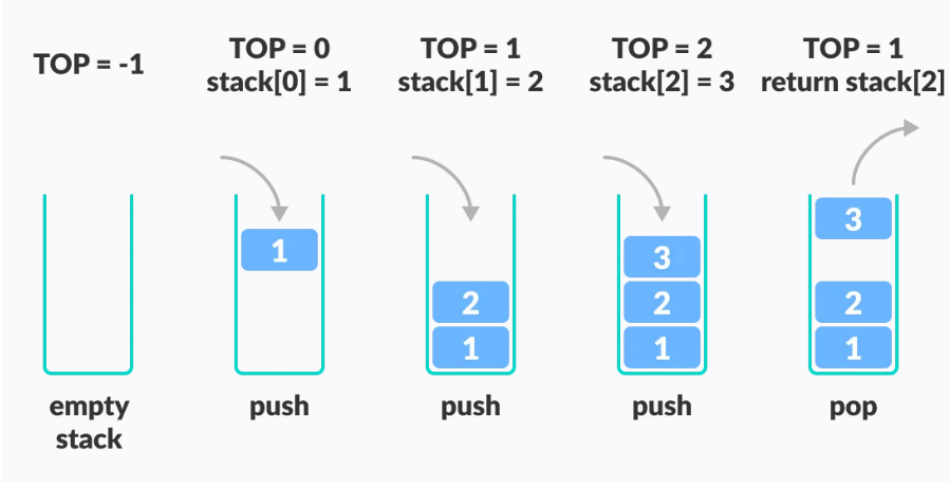
2. Dasar Teori

Sebuah stack atau tumpukan merupakan struktur data yang berfungsi untuk menyimpan dan mengelola kumpulan data dengan prinsip Last-In, First-Out (LIFO). Analogi yang sering digunakan adalah tumpukan piring di kafetaria, di mana piring terakhir yang ditambahkan akan menjadi yang pertama diambil.

Dalam implementasinya, stack dapat direpresentasikan sebagai struktur data terurut yang memiliki dua operasi utama: push dan pop. Operasi push digunakan untuk menambahkan elemen baru ke dalam stack, sementara operasi pop digunakan untuk menghapus elemen teratas dari stack.

Prinsip LIFO yang menjadi dasar stack membuatnya sangat bermanfaat dalam berbagai aplikasi, termasuk manajemen memori komputer, evaluasi ekspresi aritmatika, dan manajemen panggilan fungsi dalam pemrograman. Sebagai contoh, dalam manajemen memori, stack digunakan untuk menyimpan alamat-alamat memori yang dialokasikan untuk variabel dan fungsi.

Dengan prinsip LIFO ini, stack memungkinkan akses data dengan efisiensi, di mana elemen yang terakhir dimasukkan akan menjadi yang pertama diambil. Hal ini menjadikannya salah satu struktur data yang sangat penting dalam pengembangan perangkat lunak dan pemrograman komputer..



Working of Stack Data Structure

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 tanpa menghapusnya.
d. IsEmpty (Kosong): Memeriksa apakah tumpukan kosong atau tidak.
e. IsFull (Penuh): Memeriksa apakah tumpukan penuh atau tidak (terutama pada implementasi 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 tanpa menghapusnya.
h. Clear (Hapus Semua): Mengosongkan atau menghapus semua elemen dari tumpukan.
i. Search (Cari): Mencari keberadaan elemen tertentu dalam tumpukan.

3. Guided

```cpp
#include <iostream>
#define MAX 100

using namespace std;

class Stack {
private:
    int top;
    int arr[MAX];

public:
    Stack() { top = -1; }

    bool isFull() { return top == MAX - 1; }
    bool isEmpty() { return top == -1; }

    void push(int x) {
        if (isFull()) {
            cout << "Stack Overflow\n";
            return;
        }
        arr[++top] = x;
    }

    void pop() {
        if (isEmpty()) {
            cout << "Stack Underflow\n";
            return;
        }
        top--;
    }

    int peek() {
        if (!isEmpty()) {
            return arr[top];
        }
        cout << "Stack is empty\n";
        return -1;
    }

    void display() {
        if (isEmpty()) {
            cout << "Stack is empty\n";
            return;
        }
        for (int i = top; i >= 0; i--) {
            cout << arr[i] << " ";
        }
        cout << "\n";
    }
};

int main() {
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);

    cout << "Stack elements: ";
    s.display();

    cout << "Top element: " << s.peek() << "\n";

    s.pop();
    s.pop();
    cout << "After popping, stack elements: ";
    s.display();

    return 0;
}
```

Hasil outputnya :

```
C:\Users\ASUS\OneDrive\Dokumen\tugas smt  3\Pemograman Struktur Data 3\pertemuan7>cd
emuan7\" && g++ guided.cpp -o guided && "c:\Users\ASUS\OneDrive\Dokumen\tugas smt  3\
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10
```

Kode program :

```cpp
#include <iostream>

using namespace std;

class Node {
    public:
    int data;
    Node *next;
    Node(int value) {
        data = value;
        next = nullptr;
    }
};

class Stack {
    private:
    Node* top;
    public:
    Stack() {
        top = nullptr;
    }

    bool isEmpty() {
        return top == nullptr;
    }

    void push(int x){
        Node *newNode = new Node(x);
        newNode->next = top;
        top = newNode;
    }

    void pop(){
        if(!isEmpty()){
            cout << "Stack Underflow\n";
            return;
        }
        Node* temp = top;
        top = top->next;
        delete temp;
    }

    int peek(){
        if (!isEmpty()) {
            return top->data;
        }
        cout << "Stack is empty\n";
        return -1;
    }

    void display(){
        if (isEmpty()) {
            cout << "Stack is empty\n";
            return;
        }
        Node* current = top;
        while (current != nullptr) {
            cout << current->data << " ";
            current = current ->next;
        }
        cout << "\n";
    }
};

int main() {
    Stack s;

    s.push(10);
    s.push(20);
    s.push(30);

    cout << "Stack elements are: ";
    s.display();

    cout << "Top element is: " << s.peek() << endl;

    s.pop();
    cout << "After popping top element, stack elements are: ";
    s.display();

    return 0;

}
```

4. 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.

   Kode program :

```cpp
#include <iostream>
#include <stack>
#include <cctype> // Untuk fungsi isalnum dan tolower
using namespace std;

// Fungsi untuk mengecek apakah kalimat adalah palindrom
bool isPalindrome(const string& sentence) {
    stack<char> charStack;
    string normalized = "";

    // Normalisasi: hilangkan karakter non-alfanumerik dan ubah ke huruf kecil
    for (char ch : sentence) {
        if (isalnum(ch)) {
            normalized += tolower(ch); // Tambahkan karakter ke normalized
        }
    }

    // Masukkan setiap karakter ke dalam stack
    for (char ch : normalized) {
        charStack.push(ch);
    }

    // Bangun kembali string dari stack untuk membandingkan
    string reversed = "";
    while (!charStack.empty()) {
        reversed += charStack.top();
        charStack.pop();
    }

    // Bandingkan kalimat asli yang dinormalisasi dengan versi terbalik
    return normalized == reversed;
}

int main() {
    string input;

    // Input dari pengguna
    cout << "Masukkan kalimat: ";
    getline(cin, input);

    // Periksa apakah kalimat merupakan palindrom
    if (isPalindrome(input)) {
        cout << "Kalimat tersebut adalah: Palindrom" << endl;
    } else {
        cout << "Kalimat tersebut adalah: Bukan Palindrom" << endl;
    }

    return 0;
}
```

Hasil outputnya :

```
c:\Users\ASUS\OneDrive\Dokumen\tugas smt  3\Pemograman Struktur Data 3\pertemuan7>cd "c
emuan7\" && g++ Unguided1.cpp -o Unguided1 && "c:\Users\ASUS\OneDrive\Dokumen\tugas smt
Masukkan kalimat: ini
Kalimat tersebut adalah: Palindrom
```

Penjelasan Program ;

1. Normalisasi:
   Setiap karakter diperiksa menggunakan isalnum() untuk hanya mempertahankan karakter alfanumerik.
   Karakter diubah menjadi huruf kecil menggunakan tolower() untuk menghindari perbedaan antara huruf besar dan kecil.
2. Penggunaan Stack:
   Setiap karakter dari string hasil normalisasi dimasukkan ke dalam stack.
   Karakter diambil kembali dari stack satu per satu untuk membentuk string terbalik.
3. Perbandingan:
   String hasil normalisasi dibandingkan dengan string yang dihasilkan dari stack. Jika keduanya sama, maka kalimat adalah palindrom.

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?

Kode program :

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

// Fungsi untuk membalikkan kalimat menggunakan stack
string reverseSentence(string sentence) {
    stack<char> charStack; // Stack untuk menyimpan karakter
    string reversed = "";  // String untuk menyimpan hasil pembalikan

    // Masukkan setiap karakter ke dalam stack
    for (char ch : sentence) {
        charStack.push(ch);
    }

    // Keluarkan karakter dari stack untuk membentuk string terbalik
    while (!charStack.empty()) {
        reversed += charStack.top();
        charStack.pop();
    }

    return reversed;
}

int main() {
    string sentence;

    // Input dari pengguna
    cout << "Masukkan kata (minimal 3 kata): ";
    getline(cin, sentence);

    // Proses pembalikan kalimat
    string result = reverseSentence(sentence);

    // Tampilkan hasil
    cout << "Datastack Array:" << endl;
    cout << "Data: " << result << endl;

    return 0;
}
```

Hasil outputnya :

```
c:\Users\ASUS\OneDrive\Dokumen\tugas smt  3\Pemograman Struktur Data 3\pertemuan7>cd "c:
emuan7\" && g++ Unguided2.cpp -o Unguided2 && "c:\Users\ASUS\OneDrive\Dokumen\tugas smt
Masukkan kata (minimal 3 kata): universitas telkom purwokerto
Datastack Array:
Data: otrekowrup moklet satisrevinu
```

Penjelasan Program :

4. Struktur Stack:
   Stack digunakan untuk menyimpan setiap karakter dalam kalimat.
   Stack bekerja berdasarkan prinsip LIFO (Last In, First Out).
5. Proses Pembalikan:
   Setiap karakter dari kalimat dimasukkan ke dalam stack menggunakan perulangan for. Setelah semua karakter dimasukkan, dilakukan pembacaan dari stack (pop()), yang menghasilkan urutan terbalik.
6. Fungsi Utama:
   Fungsi reverseSentence() menerima string sebagai input dan menghasilkan string yang telah dibalik.
7. Input/Output:
   Program menerima input berupa sebuah kalimat (minimal 3 kata).
   Program mengembalikan hasil pembalikan kalimat berdasarkan urutan karakter, termasuk spasi.

Kesimpulan

Dari praktikum ini, dapat disimpulkan bahwa pemahaman tentang struktur data stack sangat penting dalam pengembangan algoritma dan pemrograman. Stack, yang beroperasi berdasarkan prinsip Last-In, First-Out (LIFO), memungkinkan elemen yang terakhir dimasukkan menjadi yang pertama dikeluarkan. Dalam praktiknya, stack digunakan untuk berbagai aplikasi, seperti manajemen memori, evaluasi ekspresi aritmatika, serta manajemen panggilan fungsi. Selama praktikum, stack diterapkan untuk memecahkan masalah palindrom dengan cara menormalkan kalimat, kemudian menggunakan stack untuk membalik urutan karakter dan membandingkan hasilnya dengan string asli yang telah dinormalisasi. Jika keduanya sama, kalimat dianggap palindrom. Selain itu, stack juga digunakan untuk membalikkan kalimat dengan cara memasukkan karakter satu per satu ke dalam stack dan kemudian mengeluarkannya (pop) untuk menghasilkan urutan terbalik. Proses ini mengilustrasikan bagaimana prinsip LIFO bekerja untuk membalikkan urutan data secara efisien. Melalui implementasi fungsi-fungsi dasar stack seperti push, pop, dan top, praktikum ini memberikan pemahaman yang lebih dalam tentang bagaimana stack dapat digunakan untuk memecahkan berbagai masalah dalam pemrograman. Secara keseluruhan, praktikum ini memperkuat pentingnya pemahaman dasar struktur data stack serta aplikasinya dalam berbagai situasi nyata, seperti pemeriksaan palindrom dan pembalikan kalimat, yang menunjukkan keefektifan stack dalam mengelola data secara efisien.