LAPORAN PRAKTIKUM

MODUL 7

STACK

Disusun Oleh:

Muhammad Ikhsan Al Hakim (2311104064)

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd., M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. 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.

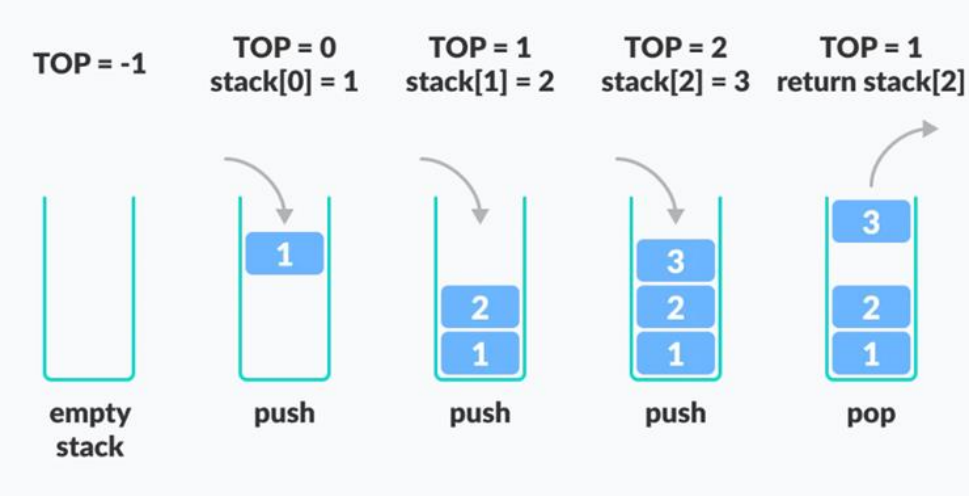
II. LANDASAN TEORI

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:

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

2. Pop (Keluarkan): Menghapus elemen dari posisi paling atas atau ujung tumpukan.
3. Top (Atas): Mendapatkan nilai atau melihat elemen teratas pada tumpukan tanpa menghapusnya
4. IsEmpty (Kosong): Memeriksa apakah tumpukan kosong atau tidak
5. IsFull (Penuh): Memeriksa apakah tumpukan penuh atau tidak (terutama pada implementasi tumpukan dengan kapasitas terbatas
6. Size (Ukuran): Mengembalikan jumlah elemen yang ada dalam tumpukan
7. Peek (Lihat): Melihat nilai atau elemen pada posisi tertentu dalam tumpukan tanpa mengapusnya
8. Clear (Hapus Semua): Mengosongkan atau menghapus semua elemen tertentu dalam tumpukan
9. Search (Cari): Mencari keberadaan elemen tertentu dalam tumpukan

III. GUIDED

1. Guided 1

```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;
}
```

Output:

```
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10
PS D:\buat struktur data\pertemuan 7>
```

2. Guided 2

```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; // Return a sentinel value
}

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

Output:

```
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10
PS D:\buat struktur data\pertemuan 7>
```

IV. UNGUIDED

1. Unguided 1

Code:

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

using namespace std;

bool isPalindrome(string str) {
  // Membuat stack
  stack<char> st;

  // Menambahkan karakter dari string ke stack
  for (char c : str) {
    st.push(c);
  }

  // Membandingkan karakter dari string dengan karakter yang dipop dari stack
  for (char c : str) {
    if (st.top() != c) {
      return false; // Bukan palindrom
    }
    st.pop(); // Menghapus karakter teratas dari stack
  }

  return true; // Palindrom
}

int main() {
  string kalimat;

  cout << "Masukan Kalimat : ";
  getline(cin, kalimat);

  if (isPalindrome(kalimat)) {
    cout << "Kalimat tersebut adalah : Palindrom" << endl;
  } else {
    cout << "Kalimat tersebut adalah : Bukan Palindrom" << endl;
  }

  return 0;
}
```

Output:

```
Masukan Kalimat : ini
Kalimat tersebut adalah : Palindrom
PS D:\buat struktur data\pertemuan 7>
```

```
Masukan Kalimat : purwokerto
Kalimat tersebut adalah : Bukan Palindrom
PS D:\buat struktur data\pertemuan 7>
```

Penjelasan:

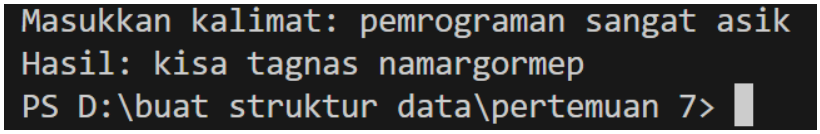
- **Deklarasi Header:** Program dimulai dengan menyertakan header **iostream** untuk input/output dan **string** untuk manipulasi string, serta **stack** untuk menggunakan struktur data stack.
- Fungsi IsPalindrom:
  - Fungsi ini menerima string str sebagai input.
  - Deklarasi Stack: Deklarasi stack st untuk menyimpan karakter dari string.
  - Penambahan Karakter ke Stack: Loop for digunakan untu-menambahkan setiap karakter dari str ke dalam stack st
  - Perbandingan Karakter: Loop for kedua digunakan untuk membandingkan karakter dari str dengan karakter teratas (top()) stack st. Jika tidak sama, maka false dikembalikan (bukan palindrom).
  - Penghapusan Karakter dari Stack: Setelah perbandingan, karakter teratas dihapus dari stack dengan st.pop().
  - Kembalikan true: Jika semua karakter cocok, fungsi mengembalikan true (palindrom)
- Fungsi main:
  - Input Kalimat: Program meminta pengguna untuk memasukkan kalimat melalui `getline(cin, kalimat)

2. Unguided 2

   Code:

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

using namespace std;

void reverseString(string& str) {
    stack<char> s;
    for (char c : str) {
        s.push(c);
    }
    str = "";
    while (!s.empty()) {
        str += s.top();
        s.pop();
    }
}

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

    // Memisahkan kata-kata dalam kalimat
    string kata;
    stack<string> kataStack;
    for (int i = 0; i < kalimat.length(); i++) {
        if (kalimat[i] != ' ') {
            kata += kalimat[i];
        } else {
            reverseString(kata);
            kataStack.push(kata);
            kata = "";
        }
    }
    // Membalikkan kata terakhir jika ada
    if (!kata.empty()) {
        reverseString(kata);
        kataStack.push(kata);
    }

    // Menggabungkan kata-kata yang telah dibalik
    string hasil = "";
    while (!kataStack.empty()) {
        hasil += kataStack.top() + " ";
        kataStack.pop();
    }

    cout << "Hasil: " << hasil << endl;

    return 0;
}
```

Output:



Penjelasan:

- Fungsi reverseString:
  - Menerima string sebagai parameter
  - Membalikkan urutan karakter dalam string menggunakan stack
  - Setiap karakter dalam string didorong ke dalam stack
  - Karakter-karakter kemudian dipop dari stack dan ditambahkan ke string yang baru
- Fungsi main:
  - Meminta pengguna untuk memasukkan kalimat
  - Memisahkan setiap kata dalam kalimat
  - Membalikkan setiap kata menggunakan fungsi reverseString
  - Menambahkan kata yang telah dibalik ke dalam stack kata
  - Menggabungkan kata-kata dari stack menjadi string hasil
  - Menampilkan hasil ke layar