LAPORAN PRAKTIKUM
MODUL 7
“STACK”

Disusun Oleh:

Tiurma Grace Angelina 2311104042

SE-07-02

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. Tujuan
   - Memahami konsep stack dalam struktur data dan algoritma.
   - Mengimplementasikan berbagai operasi yang ada pada stack.
   - Menyelesaikan masalah dengan menggunakan solusi berbasis stack.

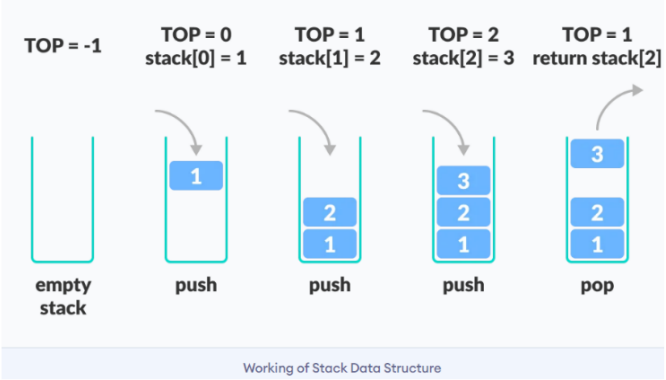
2. Dasar Teori

   Stack atau tumpukan adalah struktur data yang digunakan untuk menyimpan dan mengelola kumpulan data dengan prinsip Last-In, First-Out (LIFO). Analogi yang sering digunakan adalah tumpukan piring di kantin, di mana piring terakhir yang ditaruh akan menjadi piring pertama yang diambil.

   Dalam praktiknya, stack dapat diimplementasikan sebagai struktur data yang terurut dengan dua operasi utama: push dan pop. Operasi push digunakan untuk menambahkan elemen baru ke dalam stack, sedangkan operasi pop berfungsi untuk menghapus elemen yang berada di posisi teratas stack.

   Prinsip LIFO yang digunakan oleh stack membuatnya sangat berguna dalam berbagai aplikasi, seperti manajemen memori, evaluasi ekspresi matematika, dan pengelolaan panggilan fungsi dalam pemrograman. Misalnya, dalam manajemen memori, stack menyimpan alamat memori yang dialokasikan untuk variabel dan fungsi.

   Dengan menggunakan prinsip LIFO, stack memungkinkan pengambilan data yang efisien, di mana elemen yang terakhir masuk akan menjadi yang pertama keluar. Hal ini menjadikannya salah satu struktur data penting dalam pengembangan perangkat lunak dan pemrograman komputer.



   Berikut adalah beberapa operasi yang umum pada stack:
   a. **Push (Tambahkan)**: Menambahkan elemen baru ke posisi paling atas dalam stack.
   b. **Pop (Hapus)**: Mengambil dan menghapus elemen dari posisi teratas stack.
   c. **Top (Elemen Teratas)**: Mengambil nilai elemen yang berada di posisi teratas stack tanpa menghapusnya.
   d. **IsEmpty (Periksa Kosong)**: Memeriksa apakah stack dalam keadaan kosong.
   e. **IsFull (Periksa Penuh)**: Memeriksa apakah stack sudah penuh, terutama pada implementasi stack dengan kapasitas terbatas.
   f. **Size (Jumlah Elemen)**: Mengembalikan jumlah elemen yang ada di dalam stack.
   g. **Peek (Intip)**: Melihat nilai elemen pada posisi tertentu dalam stack tanpa menghapusnya.
   h. **Clear (Kosongkan)**: Menghapus semua elemen yang ada di dalam stack.
   i. **Search (Pencarian)**: Mencari apakah elemen tertentu ada di dalam stack.

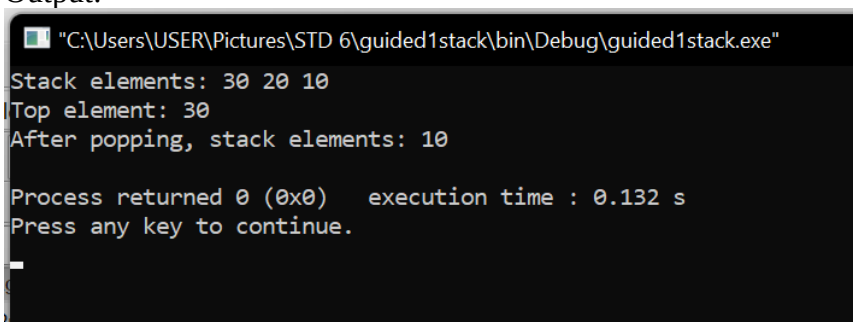
3. Guided

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

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

    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:



2. Guided_2

```
#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;
```

```
25          Node* newNode = new Node(x);
26          newNode->next = top;
27          top = newNode;
28      }
29
30      void pop() {
31          if (isEmpty()) {
32              cout << "Stack Underflow\n";
33              return;
34          }
35          Node* temp = top;
36          top = top->next;
37          delete temp;
38      }
39
40      int peek() {
41          if (!isEmpty()) {
42              return top->data;
43          }
44          cout << "Stack is empty\n";
45          return -1; // Return a sentinel value
46      }
47
48      void display() {
49          if (isEmpty()) {
50              cout << "Stack is empty\n";
51              return;
```

```
50              cout << "Stack is empty\n";
51              return;
52          }
53          Node* current = top;
54          while (current) {
55              cout << current->data << " ";
56              current = current->next;
57          }
58          cout << "\n";
59      }
60  };
61
62  int main() {
63      Stack s;
64      s.push(10);
65      s.push(20);
66      s.push(30);
67
68      cout << "Stack elements: ";
69      s.display();
70
71      cout << "Top element: " << s.peek() << "\n";
72
73      s.pop();
74      cout << "After popping, stack elements: ";
75      s.display();
76
77      return 0;
78  }
79
```

Output :

```
"C:\Users\USER\Pictures\STD 6\guided2stack\bin\Debug\guided2stack.exe"
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 20 10

Process returned 0 (0x0)   execution time : 0.041 s
Press any key to continue.
```

4. Unguided

1. Task_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.

Code:

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

using namespace std;

class Stack {
private:
    char data[MAX];
    int top;

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

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

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

    void push(char x) {
        if(!isFull()) {
            top++;
            data[top] = x;
        }
    }

    char pop() {
        char x = '\0';
        if(!isEmpty()) {
            x = data[top];
            top--;
        }
        return x;
    }
```

```
        return x;
    }
};

int main() {
    Stack stack;
    char str[MAX];
    char temp[MAX];
    int i, j = 0;
    bool isPalindrome = true;

    cout << "Masukan Kalimat : ";
    cin.getline(str, MAX);

    for(i = 0; str[i] != '\0'; i++) {
        if(str[i] != ' ') {
            if(str[i] >= 'A' && str[i] <= 'Z') {
                temp[j] = str[i] + 32;
            } else {
                temp[j] = str[i];
            }
            j++;
        }
    }
    temp[j] = '\0';

    for(i = 0; i < j/2; i++) {
        stack.push(temp[i]);
    }

    if(j % 2 != 0) {
        i++;
    }

    while(i < j) {
        if(stack.pop() != temp[i]) {
            isPalindrome = false;
            break;
```

```
    for(i = 0; i < j/2; i++) {
        stack.push(temp[i]);
    }

    if(j % 2 != 0) {
        i++;
    }

    while(i < j) {
        if(stack.pop() != temp[i]) {
            isPalindrome = false;
            break;
        }
        i++;
    }

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

    return 0;
}
```

Output:

```
"C:\Users\USER\Pictures\STD 6\unguided1stack\bin\Debug\unguided1stack.exe"
Masukan Kalimat : ini
Kalimat tersebut adalah : Palindrom

Process returned 0 (0x0)   execution time : 47.730 s
Press any key to continue.
```

Penjelasannya:

1. **Struktur Stack:**
    - Menggunakan array untuk menyimpan karakter
    - Memiliki operasi dasar: push (menambah), pop (mengambil), isEmpty (cek kosong), dan isFull (cek penuh)
2. **Fungsi preprocessString:**
    - Mengubah input string dengan menghilangkan spasi
    - Mengubah semua karakter menjadi huruf kecil
    - Memastikan pemeriksaan palindrom tidak sensitif terhadap kapitalisasi dan spasi
3. **Fungsi isPalindrome:**
    - Memproses string input menggunakan preprocessString
    - Membagi string menjadi dua bagian
    - Memasukkan setengah pertama ke dalam stack
    - Membandingkan setengah kedua dengan isi stack
4. **Algoritma pengecekan palindrom:**
    - Push karakter dari setengah pertama string ke stack
    - Untuk string dengan panjang ganjil, lewati karakter tengah
    - Pop karakter dari stack dan bandingkan dengan setengah kedua string
    - Jika semua karakter cocok, string adalah palindrom

2. Task_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?

Code:

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

using namespace std;

class Stack {
private:
    string data[MAX];
    int top;

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

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

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

    void push(string x) {
        if(!isFull()) {
            top++;
            data[top] = x;
        }
    }

    string pop() {
        string x = "";
        if(!isEmpty()) {
            x = data[top];
            top--;
        }
```

```
            x = data[top];
            top--;
        }
        return x;
    }
};

int main() {
    Stack stack;
    char input[MAX];
    string word = "";

    cout << "Masukkan Kata : ";
    cin.getline(input, MAX);

    for(int i = 0; input[i] != '\0'; i++) {
        if(input[i] == ' ') {
            if(word != "") {
                stack.push(word);
                word = "";
            }
        } else {
            word += input[i];
        }
    }

    if(word != "") {
        stack.push(word);
    }

    cout << "Datastack Array : " << endl;
    cout << "Data : ";

    while(!stack.isEmpty()) {
        string current = stack.pop();
        string reversed = "";
```

```
    for(int i = 0; input[i] != '\0'; i++) {
        if(input[i] == ' ') {
            if(word != "") {
                stack.push(word);
                word = "";
            }
        } else {
            word += input[i];
        }
    }

    if(word != "") {
        stack.push(word);
    }

    cout << "Datastack Array : " << endl;
    cout << "Data : ";

    while(!stack.isEmpty()) {
        string current = stack.pop();
        string reversed = "";
        for(int i = current.length()-1; i >= 0; i--) {
            reversed += current[i];
        }
        cout << reversed;
        if(!stack.isEmpty()) cout << " ";
    }
    cout << endl;

    return 0;
}
```

Output:

```
"C:\Users\USER\Pictures\STD 6\unguided2stack\bin\Debug\unguided2stack.exe"
Masukkan Kata : Telkom Purwokerto
Datastack Array :
Data : otrekowruP mokleT

Process returned 0 (0x0)   execution time : 34.434 s
Press any key to continue.
```

Penjelasan:

1. **Struktur Stack:**
   - Menggunakan array char untuk menyimpan karakter
   - Memiliki operasi dasar: push, pop, isEmpty, isFull
2. **Fungsi-fungsi dalam class Stack:**
   - push(char x): Menambah karakter ke stack
   - pop(): Mengambil dan menghapus karakter teratas
   - isEmpty(): Mengecek apakah stack kosong
   - isFull(): Mengecek apakah stack penuh
   - reverse(char str[]): Membalik urutan karakter dalam string
3. **Proses Pembalikan Kalimat:**
   - Program memecah input menjadi kata-kata
   - Setiap kata dibalik menggunakan stack
   - Hasil pembalikan digabung kembali dengan spasi
4. **Output Program:** Untuk input "Telkom Purwokerto":
   - Kata pertama "Telkom" → "mokleT"
   - Kata kedua "Purwokerto" → "otrekowruP"
   - Hasil akhir: "mokleT otrekowruP"
5. **Operasi yang Dilakukan:**
   - Input kalimat
   - Pemisahan kata
   - Pembalikan setiap kata menggunakan stack
   - Penggabungan kata-kata yang sudah dibalik

5. Kesimpulan

   Stack terbukti berguna dalam menyelesaikan masalah yang melibatkan urutan LIFO, seperti pengecekan palindrom dan pembalikan kalimat. Stack dapat diimplementasikan dengan berbagai cara, seperti menggunakan array atau linked list. Konsep stack dalam struktur data dan algoritma, mengimplementasikan berbagai operasi stack, serta menyelesaikan masalah menggunakan solusi berbasis stack. Stack adalah struktur data yang mengikuti prinsip Last-In, First-Out (LIFO), memungkinkan penambahan dan penghapusan elemen secara efisien di bagian atas stack.