

**LAPORAN PRAKTIKUM**  
**Modul 7**  
**“STACK”**



**Disusun Oleh:**

Ahmad Al - Farizi - 2311104054

**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.

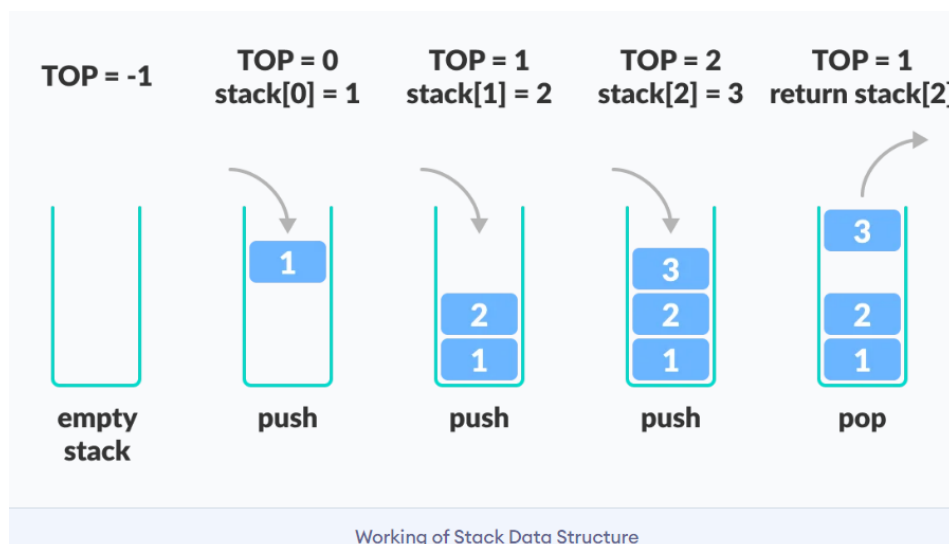
## 2. Landasan Teori

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.



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 menghapusnya.
8. Clear (Hapus Semua): Mengosongkan atau menghapus semua elemen dari tumpukan.
9. Search (Cari): Mencari keberadaan elemen tertentu dalam tumpukan

### **3. Guided**

#### **1. Guided 7.1**

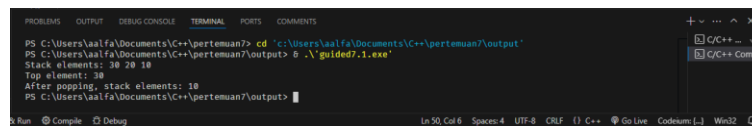
Program tersebut mengimplementasikan Stack berbasis array dengan kapasitas maksimum 100 elemen. Stack adalah struktur data “LIFO (Last In, First Out)” di mana elemen terakhir yang dimasukkan akan dikeluarkan terlebih dahulu. Kelas Stack memiliki atribut `top` untuk menunjukkan elemen teratas dan `arr[MAX]` sebagai tempat penyimpanan. Operasi utama yang tersedia meliputi `push()` untuk menambah elemen, `pop()` untuk menghapus elemen teratas, `peek()` untuk melihat elemen teratas tanpa menghapusnya, dan `display()` untuk mencetak seluruh elemen stack. Fungsi `isFull()` dan `isEmpty()` digunakan untuk memeriksa apakah stack penuh atau kosong. Program utama mendemonstrasikan penambahan, penghapusan, dan pencetakan elemen, menunjukkan cara kerja stack dalam menyimpan dan mengelola data secara efisien.

Kode Program:

```
1  #include <iostream>
2  #define MAX 100
3
4  using namespace std;
5
6  class Stack {
7  private:
8      int top;
9      int arr[MAX];
10
11 public:
12     Stack() { top = -1; }
13
14     bool isFull() { return top == MAX - 1; }
15     bool isEmpty() { return top == -1; }
16
17     void push(int x) {
18         if (isFull()) {
19             cout << "Stack Overflow\n";
20             return;
21         }
22         arr[++top] = x;
23     }
24
25     void pop() {
26         if (isEmpty()) {
27             cout << "Stack Underflow\n";
28             return;
29         }
30         top--;
31     }
32
33     int peek() {
34         if (!isEmpty()) {
35             return arr[top];
36         }
37         cout << "Stack is empty\n";
38         return -1;
39     }
40
41     void display() {
42         if (isEmpty()) {
43             cout << "Stack is empty\n";
44             return;
45         }
46         for (int i = top; i >= 0; i--) {
47             cout << arr[i] << " ";
48         }
49         cout << "\n";
50     }
51 };
52
```

```
1  int main() {
2      Stack s;
3      s.push(10);
4      s.push(20);
5      s.push(30);
6
7      cout << "Stack elements: ";
8      s.display();
9
10     cout << "Top element: " << s.peek() << "\n";
11
12     s.pop();
13     s.pop();
14     cout << "After popping, stack elements: ";
15     s.display();
16
17     return 0;
18 }
```

Output dari Kode Program:



```
PS C:\Users\laalfa\Documents\C++\pertemuan7> cd 'c:\Users\laalfa\Documents\C++\pertemuan7\output'
PS C:\Users\laalfa\Documents\C++\pertemuan7\output> g++ -o guided7.1.exe
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10
PS C:\Users\laalfa\Documents\C++\pertemuan7\output>
```

## 2. Guided 7.2

Program ini mengimplementasikan “Stack” menggunakan “linked list” untuk memanfaatkan alokasi memori dinamis tanpa batasan ukuran tetap. Kelas Node merepresentasikan elemen stack dengan atribut data untuk menyimpan nilai dan next untuk menunjuk elemen berikutnya. Kelas Stack memiliki operasi utama, yaitu push() untuk menambah elemen di atas stack, pop() untuk menghapus elemen teratas, peek() untuk melihat elemen teratas tanpa menghapusnya, dan display() untuk mencetak semua elemen dari atas ke bawah. Kondisi stack kosong diperiksa dengan isEmpty(). Dalam fungsi utama, program menambahkan elemen 10, 20, dan 30, mencetak elemen stack, menampilkan elemen teratas, lalu menghapus dua elemen. Implementasi ini lebih fleksibel dibandingkan stack berbasis array karena tidak dibatasi ukuran tetap, tetapi memerlukan pengelolaan memori tambahan.

Kode Program :

```
1  #include <iostream>
2
3  using namespace std;
4
5  class Node {
6  public:
7      int data;
8      Node *next;
9      Node(int value) {
10         data = value;
11         next = nullptr;
12     }
13 };
14
15 class Stack {
16 private:
17     Node* top;
18 public:
19     Stack() {
20         top = nullptr;
21     }
22
23     bool isEmpty() {
24         return top == nullptr;
25     }
26
27     void push(int x){
28         Node *newNode = new Node(x);
29         newNode->next = top;
30         top = newNode;
31     }
32
33     void pop(){
34         if(!isEmpty()){
35             cout << "Stack Underflow\n";
36             return;
37         }
38         Node* temp = top;
39         top = top->next;
40         delete temp;
41     }
42
43     int peek(){
44         if (!isEmpty()) {
45             return top->data;
46         }
47         cout << "Stack is empty\n";
48         return -1;
49     }
```

```
1 void display(){
2     if (isEmpty()) {
3         cout << "Stack is empty\n";
4         return;
5     }
6     Node* current = top;
7     while (current != nullptr) {
8         cout << current->data << " ";
9         current = current->next;
10    }
11    cout << "\n";
12 }
13 };
14
15 int main() {
16     Stack s;
17     s.push(10);
18     s.push(20);
19     s.push(30);
20
21     cout << "Stack elements: ";
22     s.display();
23
24     cout << "Top element: " << s.peek() << "\n";
25
26     s.pop();
27     s.pop();
28     cout << "After popping, stack elements: ";
29     s.display();
30
31     return 0;
32 }
```

Output dari Kode Program:



```
PS C:\Users\laifa\Documents\C++\pertemuan7\output> g++ guided7.2.exe
Stack elements: 30 20 10
Top element: 30
Stack Underflow
Stack Underflow
After popping, stack elements: 30 20 10
PS C:\Users\laifa\Documents\C++\pertemuan7\output>
```

#### 4. Unguided

##### 1. Soal No 1

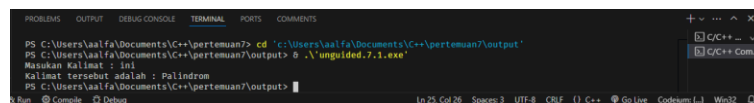
Kode program di bawah memeriksa apakah sebuah string adalah palindrom dengan mengabaikan huruf besar/kecil dan karakter non-alfanumerik. Fungsi `isPalindrome` membersihkan input menggunakan loop untuk menyaring hanya karakter alfanumerik dan mengonversinya ke huruf kecil. Karakter ini kemudian dimasukkan ke dalam stack untuk memanfaatkan sifat “LIFO (Last In, First Out)”. Proses pengecekan dilakukan dengan

membandingkan setiap karakter string yang dibersihkan dengan karakter yang dikeluarkan dari stack. Jika seluruh pasangan cocok, string dianggap palindrom. Pada fungsi main, input dari pengguna diproses oleh `isPalindrome`. Program mencetak hasil apakah string tersebut termasuk palindrom atau tidak, menggunakan logika yang sederhana dan efisien.

Kode Program:

```
1  #include <iostream>
2  #include <stack>
3  #include <string>
4  #include <algorithm>
5
6  bool isPalindrome(const std::string& str) {
7      std::stack<char> stack;
8      std::string cleaned = "";
9
10     for (char c : str) {
11         if (std::isalnum(c)) {
12             cleaned += std::tolower(c);
13         }
14     }
15
16     for (char c : cleaned) {
17         stack.push(c);
18     }
19
20     for (char c : cleaned) {
21         if (stack.top() != c) {
22             return false;
23         }
24         stack.pop();
25     }
26
27     return true;
28 }
29
30 int main() {
31     std::string input;
32     std::cout << "Masukan Kalimat : ";
33     std::getline(std::cin, input);
34
35     if (isPalindrome(input)) {
36         std::cout << "Kalimat tersebut adalah : Palindrom\n";
37     } else {
38         std::cout << "Kalimat tersebut adalah : Bukan Palindrom\n";
39     }
40
41     return 0;
42 }
43
```

Output dari Kode Program:



```
PS C:\Users\jaalifa\Documents\C++\pertemuan7> cd 'C:\Users\jaalifa\Documents\C++\pertemuan7\output'
PS C:\Users\jaalifa\Documents\C++\pertemuan7\output> B .\unguided.7.1.exe
Masukan Kalimat : ini
Kalimat tersebut adalah : Palindrom
PS C:\Users\jaalifa\Documents\C++\pertemuan7\output>
```



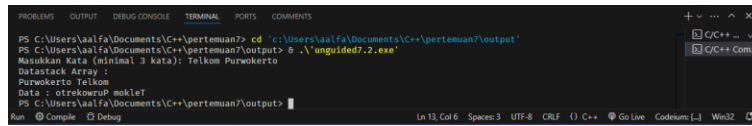
## 2. Soal No 2

Program ini membalik urutan kata dalam kalimat yang dimasukkan pengguna, lalu membalik setiap huruf dalam kata tersebut. Pada fungsi main, kalimat diinput dan diproses oleh fungsi reverseWords. Dalam reverseWords, setiap kata dimasukkan ke dalam stack menggunakan std::stringstream dan stack.push. Program mencetak isi stack untuk menunjukkan urutan kata yang terakhir masuk di atas (Datastack Array). Selanjutnya, setiap kata dikeluarkan dari stack, hurufnya dibalik dengan std::reverse, lalu dicetak ke layar. Struktur stack digunakan untuk membalik urutan kata secara efisien, sementara std::reverse menangani pembalikan huruf dalam kata. Output akhirnya berupa kata – kata dalam urutan terbalik dengan huruf dalam tiap kata juga terbalik.

Kode Program:

```
1  #include <iostream>
2  #include <stack>
3  #include <sstream>
4  #include <algorithm>
5
6  void reverseWords(const std::string& str) {
7      std::stack<std::string> stack;
8      std::stringstream ss(str);
9      std::string word;
10
11     while (ss >> word) {
12         stack.push(word);
13     }
14
15     std::cout << "Datastack Array :\n";
16
17     std::stack<std::string> tempStack = stack;
18     while (!tempStack.empty()) {
19         std::cout << tempStack.top() << " ";
20         tempStack.pop();
21     }
22     std::cout << "\n";
23
24     std::cout << "Data : ";
25     while (!stack.empty()) {
26         std::string reversedWord = stack.top();
27         std::reverse(reversedWord.begin(), reversedWord.end());
28         std::cout << reversedWord << " ";
29         stack.pop();
30     }
31     std::cout << std::endl;
32 }
33
34 int main() {
35     std::string input;
36     std::cout << "Masukkan Kata (minimal 3 kata): ";
37     std::getline(std::cin, input);
38
39     reverseWords(input);
40
41     return 0;
42 }
43
```

### Output dari Kode Program:



```
PS C:\Users\aaifa\Documents\C++\pertemuan7> cd "c:\Users\aaifa\Documents\C++\pertemuan7\output"
PS C:\Users\aaifa\Documents\C++\pertemuan7\output> g++ -std=c++11 *.cpp -o unguided7.2.exe
Masukkan Kata (minimal 3 kata): Telkom Purwokerto
Datastack Array :
Purwokerto Telkom
Data : otrekourup moklet
PS C:\Users\aaifa\Documents\C++\pertemuan7\output>
```

## 5. Kesimpulan

Stack adalah struktur data penting yang bekerja dengan prinsip Last – In, First – Out (LIFO), di mana elemen terakhir yang dimasukkan akan menjadi yang pertama diambil. Dengan operasi utama push untuk menambahkan elemen dan pop untuk menghapus elemen teratas, stack memungkinkan pengelolaan data yang efisien. Prinsip ini menjadikan stack sangat berguna dalam berbagai aplikasi, seperti manajemen memori, evaluasi ekspresi aritmatika, dan pengelolaan panggilan fungsi. Efisiensi dan kemudahan akses data membuat stack menjadi salah satu elemen fundamental dalam pengembangan perangkat lunak dan pemrograman.

