

**LAPORAN PRAKTIKUM**  
**Stack**  
**Bagian Pertama**



**Disusun Oleh:**

**Ryan Gabriel Togar Simamora (2311104045)**

**Kelas : SE0702**

**Dosen :**

**Wahyu Andi Saputra**

**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

### A. Stack

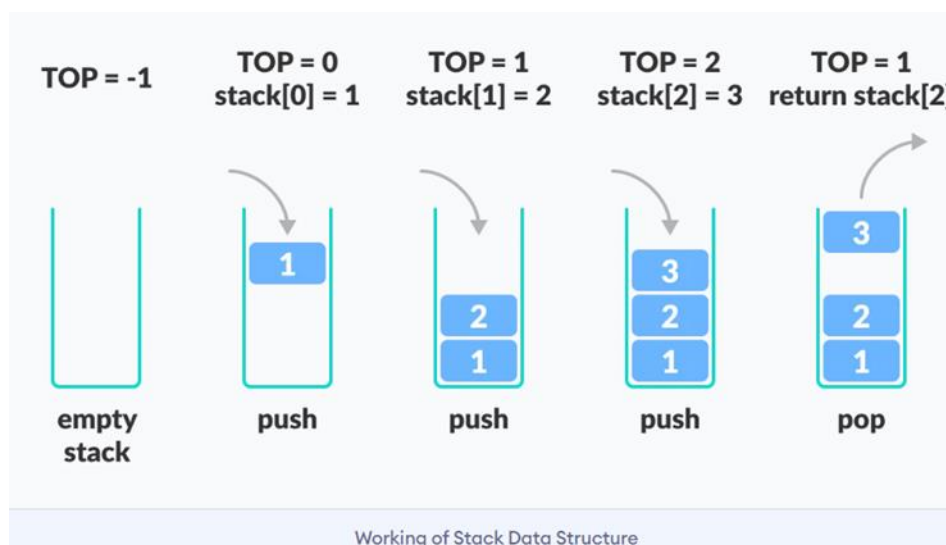
Apa itu Stack?

Stack atau tumpukan adalah struktur data yang bekerja dengan prinsip LIFO (Last In, First Out). Artinya, elemen yang terakhir dimasukkan akan menjadi yang pertama dikeluarkan, mirip dengan tumpukan piring. Piring terakhir yang ditaruh di atas akan menjadi piring pertama yang diambil.

### Fungsi Stack dalam Pemrograman:

Stack digunakan dalam berbagai aplikasi, seperti:

- ❖ Manajemen Memori: Menyimpan alamat-alamat memori untuk variabel atau fungsi.
- ❖ Evaluasi Ekspresi: Membantu menghitung ekspresi matematika atau logika.
- ❖ Manajemen Fungsi: Melacak urutan panggilan fungsi dalam pemrograman.



### Operasi Dasar pada Stack:

- ❖ Push (Masukkan): Menambahkan elemen ke posisi teratas stack.
- ❖ Pop (Keluarkan): Menghapus elemen dari posisi teratas stack.
- ❖ Top (Atas): Melihat elemen teratas stack tanpa menghapusnya.
- ❖ IsEmpty (Kosong): Mengecek apakah stack kosong.
- ❖ IsFull (Penuh): Mengecek apakah stack penuh (biasanya untuk stack dengan kapasitas terbatas).
- ❖ Size (Ukuran): Mengembalikan jumlah elemen dalam stack.
- ❖ Peek (Lihat): Melihat elemen pada posisi tertentu tanpa menghapusnya.
- ❖ Clear (Hapus Semua): Mengosongkan semua elemen dalam stack.
- ❖ Search (Cari): Mencari elemen tertentu dalam stack.

## Kenapa Stack Penting?

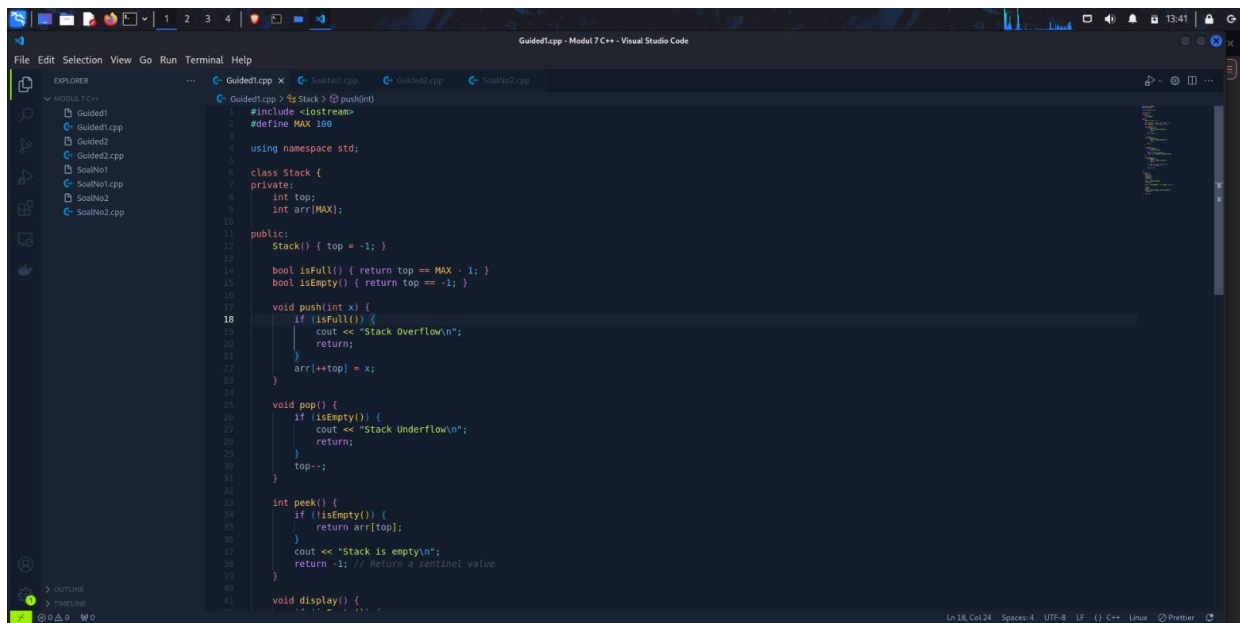
Stack memungkinkan pengelolaan data yang efisien karena hanya bekerja pada elemen paling atas. Ini membuatnya cocok untuk proses yang membutuhkan urutan tertentu, seperti memutar balik data, menyimpan histori, atau memanipulasi data sementara dalam fungsi.

## Kesimpulan:

Stack adalah salah satu struktur data yang sangat penting dalam pemrograman karena sifatnya yang sederhana namun sangat berguna. Operasi-operasi seperti push, pop, dan peek mempermudah pengelolaan data dengan prinsip LIFO. Stack banyak diterapkan dalam pengembangan perangkat lunak, terutama untuk memecahkan masalah yang melibatkan urutan dan pengelolaan data sementara.

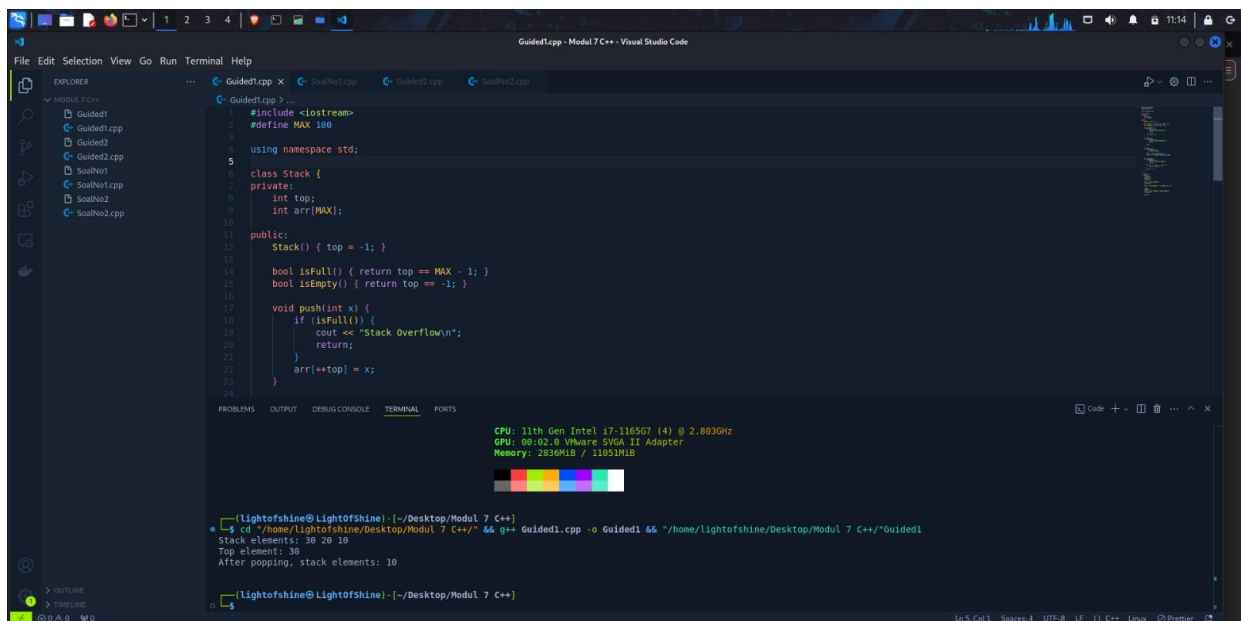
## III. Guided

### File Guided1.cpp



```
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; // Return a sentinel value
39     }
40
41     void display() {
```

### Outputnya



```
CPU: 11th Gen Intel i7-11650T (4) @ 2.803GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 2336MiB / 1103MiB

(Lightofshine@Lightofshine) ~ - /Desktop/Modul 7 C++
$ cd ~/home/Lightofshine/Desktop/Modul 7 C++/ && g++ Guided1.cpp -o Guided1 && ./Guided1
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10

(Lightofshine@Lightofshine) ~ - /Desktop/Modul 7 C++
$
```

Untuk Source Codenya Lebih Lengkap dibawah ini :

### **Guided1.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; // 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;
}

```

Penjelasan Code :

Konsep yang Ditampilkan:

1. Push: Menambahkan elemen ke stack.
2. Pop: Menghapus elemen teratas dari stack.
3. Peek: Mengakses elemen teratas tanpa menghapusnya.
4. Display: Menampilkan semua elemen stack.
5. Overflow dan Underflow: Menangani kasus ketika stack penuh atau kosong.

Stack ini mengikuti prinsip LIFO (Last In, First Out), di mana elemen terakhir yang dimasukkan adalah yang pertama kali dihapus.

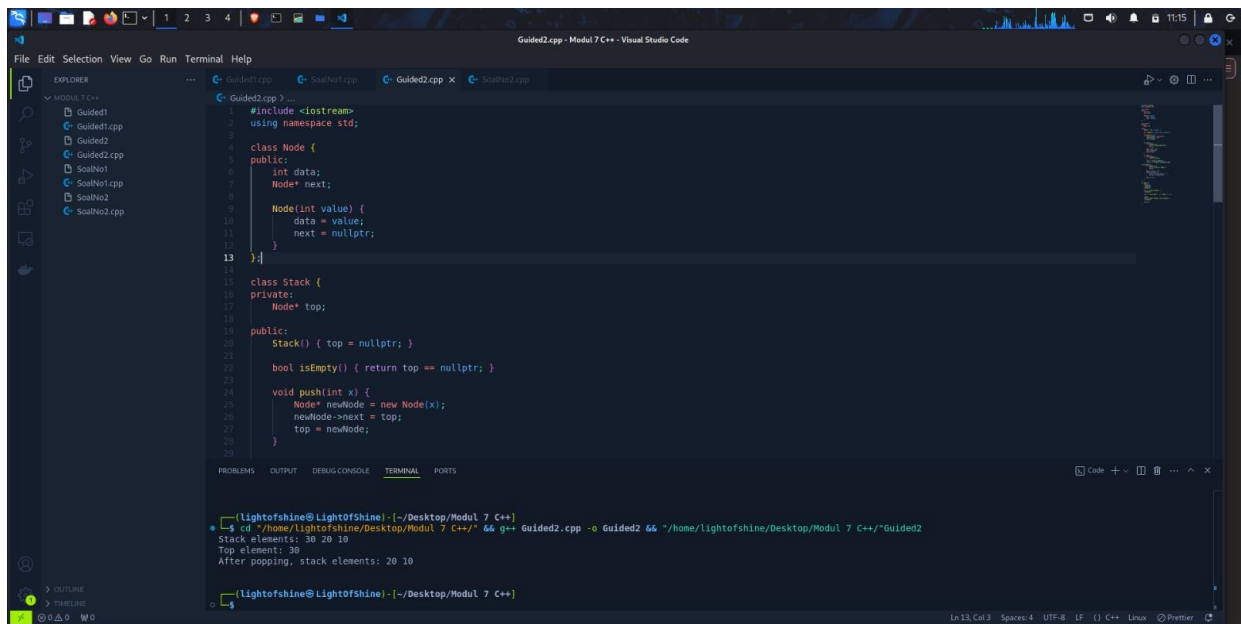
### File Guided2.cpp

```

1  #include <iostream>
2  using namespace std;
3
4  class Node {
5  public:
6      int data;
7      Node* next;
8
9      Node(int value) {
10         data = value;
11         next = nullptr;
12     }
13 };
14
15 class Stack {
16 private:
17     Node* top;
18
19 public:
20     Stack() { top = nullptr; }
21
22     bool isEmpty() { return top == nullptr; }
23
24     void push(int x) {
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()) {

```

## Outputnya



```
#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;
    }
};

int main() {
    Stack s;
    s.push(30);
    s.push(20);
    cout << "Stack elements: 30 20 10\n";
    s.pop();
    cout << "Top element: 30\n";
    s.pop();
    cout << "After popping, stack elements: 20 10\n";
    return 0;
}
```

Untuk Source Codenya Lebih Lengkap ada di bawah ini :

### Guided2.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();
    cout << "After popping, stack elements: ";
    s.display();

    return 0;
}

```

Penjelasan Code :

### **Perbedaan dengan Codingan Sebelumnya**

#### **Struktur Data:**

Codingan yang pertama menggunakan array; sedangkan codingan kedua menggunakan linked list.

#### **Dinamis vs Statis:**

Linked list memungkinkan stack memiliki ukuran dinamis, sedangkan array memiliki ukuran tetap (MAX).

#### **Manajemen Memori:**

Linked list menggunakan new dan delete untuk mengelola memori secara dinamis.

Array mengalokasikan memori secara statis di awal.

## Keunggulan Implementasi Linked List

Tidak ada batasan kapasitas seperti array.

Memori dialokasikan sesuai kebutuhan, lebih efisien untuk kasus dengan ukuran stack yang berubah-ubah.

## Kekurangan Implementasi Linked List

Memerlukan lebih banyak memori karena menyimpan pointer tambahan (next) untuk setiap elemen.

Operasi sedikit lebih lambat dibandingkan array karena memerlukan alokasi/dealokasi memori dinamis.

## IV. 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.

Contoh:

Kalimat : ini

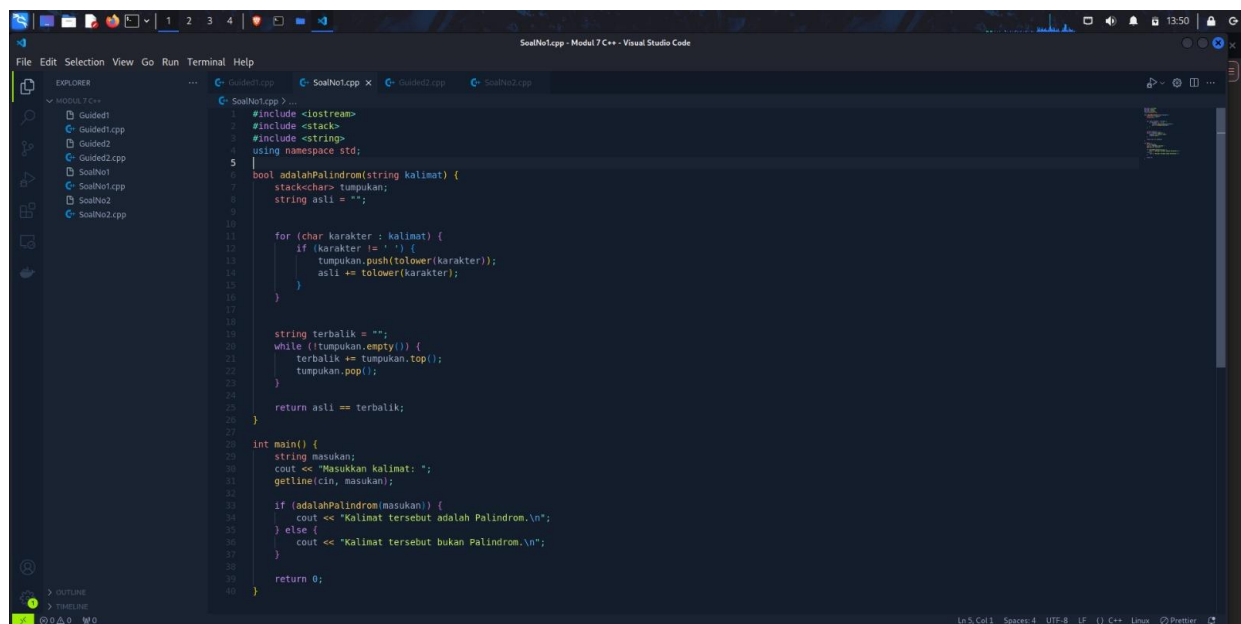
Kalimat tersebut adalah polindrom

Kalimat : telkom

Kalimat tersebut adalah bukan polindrom

```
Masukan Kalimat : ini
Kalimat tersebut adalah : Palindrom
```

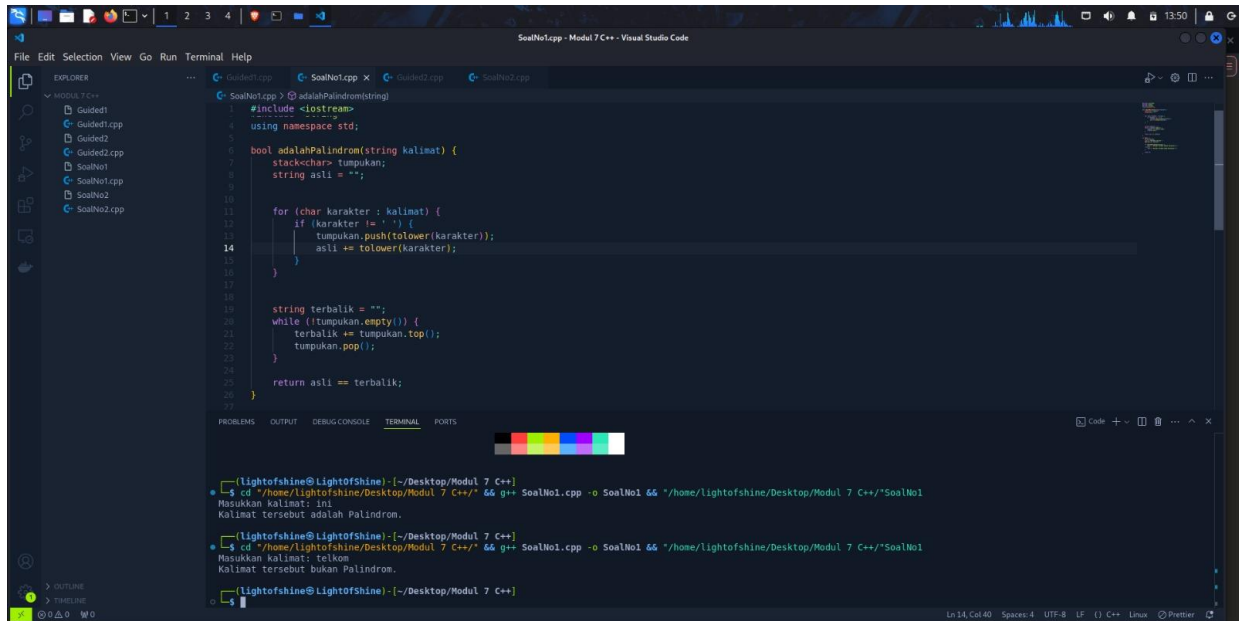
Jawab :



```
1 #include <iostream>
2 #include <stack>
3 #include <string>
4 using namespace std;
5
6 bool adalahPalindrom(string kalimat) {
7     stack<char> tumpukan;
8     string asli = "";
9
10
11     for (char karakter : kalimat) {
12         if (karakter != ' ') {
13             tumpukan.push(tolower(karakter));
14             asli += tolower(karakter);
15         }
16     }
17
18     string terbalik = "";
19     while (!tumpukan.empty()) {
20         terbalik += tumpukan.top();
21         tumpukan.pop();
22     }
23
24     return asli == terbalik;
25 }
26
27
28 int main() {
29     string masukan;
30     cout << "Masukkan kalimat: ";
31     getline(cin, masukan);
32
33     if (adalahPalindrom(masukan)) {
34         cout << "Kalimat tersebut adalah Palindrom.\n";
35     } else {
36         cout << "Kalimat tersebut bukan Palindrom.\n";
37     }
38
39     return 0;
40 }
```



Outputnya :



```
1 #include <iostream>
2 using namespace std;
3
4 bool adalahPalindrom(string kalimat) {
5     stack<char> tumpukan;
6     string asli = "";
7
8     for (char karakter : kalimat) {
9         if (karakter != ' ') {
10             tumpukan.push(tolower(karakter));
11             asli += tolower(karakter);
12         }
13     }
14
15     string terbalik = "";
16     while (!tumpukan.empty()) {
17         terbalik += tumpukan.top();
18         tumpukan.pop();
19     }
20
21     return asli == terbalik;
22 }
```

```
Lightofshine@Lightofshine:~/Desktop/Modul 7 C++$ cd ~/home/Lightofshine/Desktop/Modul 7 C++/ 66 g++ SoalNo1.cpp -o SoalNo1 66 ~/home/Lightofshine/Desktop/Modul 7 C++/SoalNo1
Masukkan kalimat: ini
Kalimat tersebut adalah Palindrom.

Lightofshine@Lightofshine:~/Desktop/Modul 7 C++$ cd ~/home/Lightofshine/Desktop/Modul 7 C++/ 66 g++ SoalNo1.cpp -o SoalNo1 66 ~/home/Lightofshine/Desktop/Modul 7 C++/SoalNo1
Masukkan kalimat: telkom
Kalimat tersebut bukan Palindrom.
```

### Penjelasan Cara kerjanya :

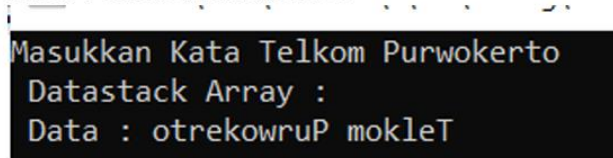
Program ini menghilangkan spasi dan mengubah semua karakter menjadi huruf kecil, kemudian menyimpan karakter-karakter tersebut dalam stack. Setelah itu, program membalikkan urutan karakter dengan mengambilnya dari stack dan membandingkan hasilnya dengan string asli untuk memeriksa apakah kalimat tersebut 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?

Contoh

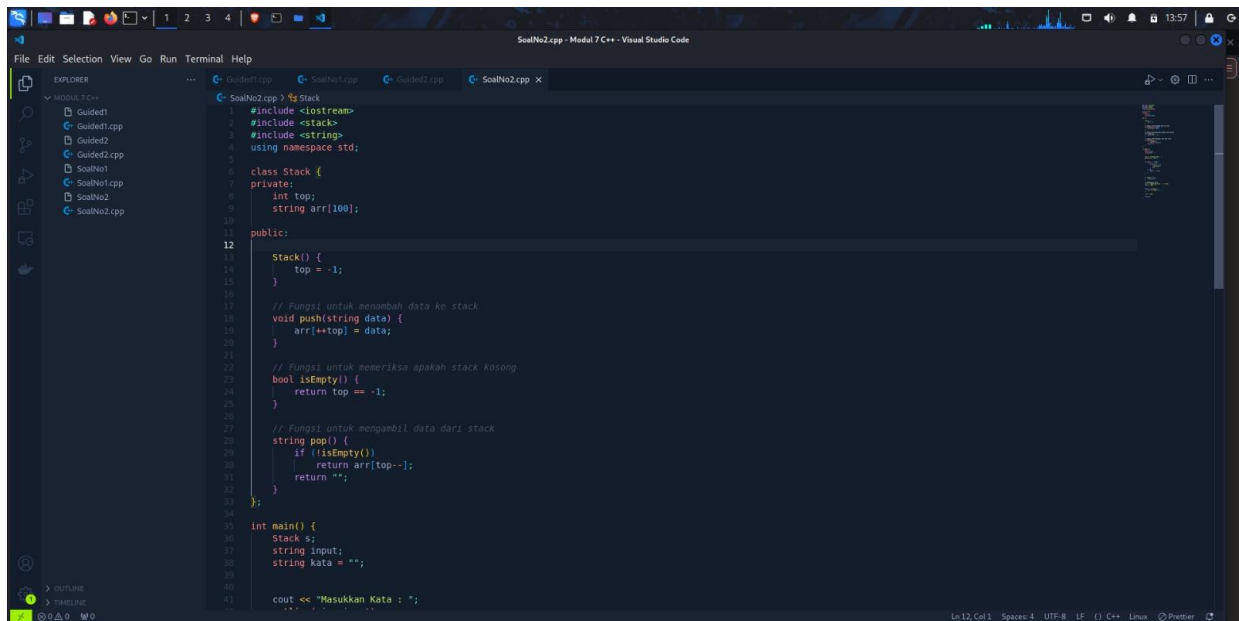
Kalimat : Telkom Purwokerto

Hasil : otrekowruP mokleT



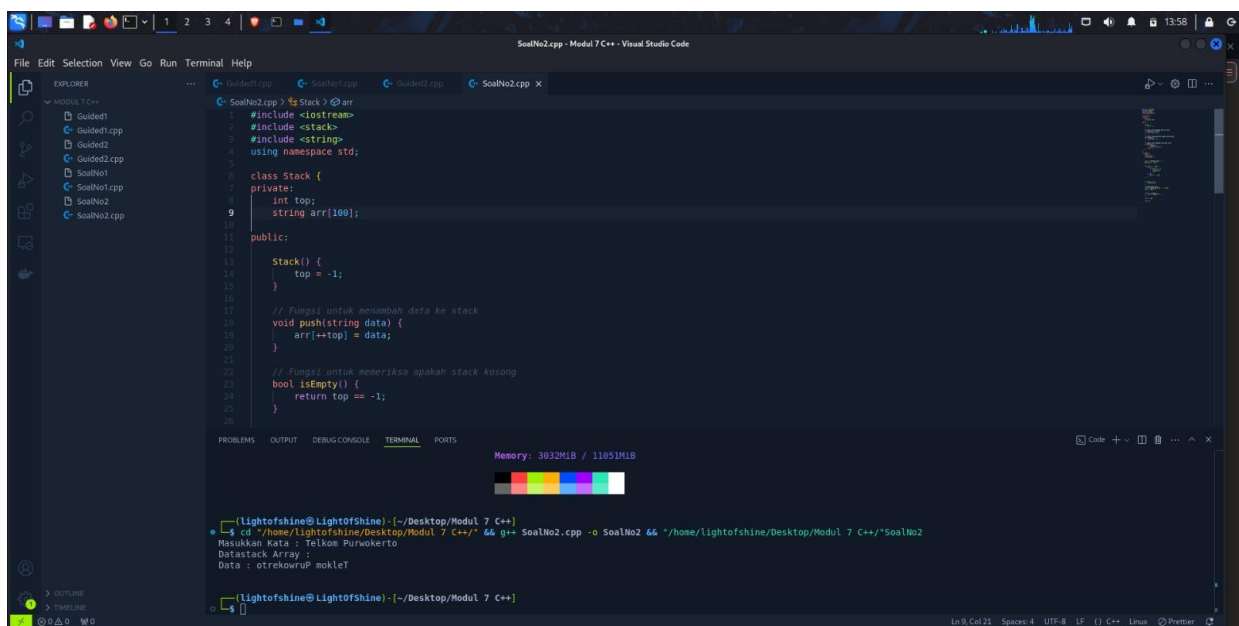
```
Masukkan Kata Telkom Purwokerto
Datastack Array :
Data : otrekowruP mokleT
```

Jawab :



```
1 #include <iostream>
2 #include <stack>
3 #include <string>
4 using namespace std;
5
6 class Stack {
7 private:
8     int top;
9     string arr[100];
10
11 public:
12     Stack() {
13         top = -1;
14     }
15
16     // Fungsi untuk menambah data ke stack
17     void push(string data) {
18         arr[++top] = data;
19     }
20
21     // Fungsi untuk memeriksa apakah stack kosong
22     bool isEmpty() {
23         return top == -1;
24     }
25
26     // Fungsi untuk mengambil data dari stack
27     string pop() {
28         if (!isEmpty())
29             return arr[top--];
30         return "";
31     }
32 };
33
34 int main() {
35     Stack s;
36     string input;
37     string kata = "";
38
39     cout << "Masukkan Kata : ";
40     while (getchar() != '\n')
41         continue;
```

Outputnya :



```
Memory: 3832MB / 11851MB
(Lightofshine@Lightofshine) ~ - [~/Desktop/Modul 7 C++]
$ cd ~/home/Lightofshine/Desktop/Modul 7 C++/ && g++ SoalNo2.cpp && ./SoalNo2
Masukkan Kata : Telkom Purwokerto
Data : otrekowurp mogleT
$
```

Penjelasan :

**Fungsi/Operasi yang Dibuat :**

push(string data): Menambahkan kata ke dalam stack.

isEmpty(): Memeriksa apakah stack kosong.

pop(): Mengambil kata dari stack (dalam urutan terbalik).

**Ringkasan**

Program ini menggunakan stack untuk membalikkan urutan kata dalam kalimat yang dimasukkan oleh pengguna. Setiap kata dimasukkan ke dalam stack, dan kemudian kata-kata tersebut dikeluarkan dan ditampilkan kembali dalam urutan terbalik.

**Penjelasan Proses:**

Memecah Input Menjadi Kata-kata:

Program akan membaca karakter demi karakter dari input.

Ketika menemukan spasi (' '), program menganggapnya sebagai pemisah kata dan akan menyimpan kata yang sudah terbentuk di dalam stack.

Dalam hal ini, program akan memecah input menjadi dua kata:

Kata pertama: "Telkom"

Kata kedua: "Purwokerto"

Memasukkan Kata ke dalam Stack:

Setiap kata dimasukkan ke dalam stack dengan cara dibalik karakter-karakternya.

Kata pertama yang dimasukkan adalah "Telkom". Sebelum dimasukkan, karakter-karakternya dibalik menjadi "mokleT", dan dimasukkan ke dalam stack.

Kata kedua yang dimasukkan adalah "Purwokerto". Karakter-karakternya dibalik menjadi "otrekowruP", dan dimasukkan ke dalam stack.

Stack Setelah Pemrosesan: Setelah memecah dan membalik karakter dari setiap kata, stack berisi:

Stack: otrekowruP, mokleT

Urutan ini adalah urutan kata yang terbalik, tetapi karakter-karakternya juga sudah dibalik.

Menampilkan Output: Program kemudian mengambil elemen dari stack satu per satu (dengan fungsi pop()) dan menampilkan kata-kata tersebut dalam urutan terbalik dan dengan karakter-karakter yang dibalik.

### **Output yang Dihasilkan**

Masukkan Kata : Telkom Purwokerto

Datastack Array :

Data : otrekowruP mokleT

### **Penjelasan Output:**

otrekowruP adalah hasil pembalikan karakter dari kata "Purwokerto".

mokleT adalah hasil pembalikan karakter dari kata "Telkom".

Program membalik karakter dalam setiap kata dan juga membalikkan urutan kata, sehingga hasil akhirnya adalah kata-kata tersebut muncul dalam urutan terbalik dan dengan karakter-karakternya yang juga dibalik.

### **Kesimpulan**

Program ini membalikkan urutan karakter dalam setiap kata yang dimasukkan dan juga membalikkan urutan kata-kata itu sendiri. Dengan demikian, input "Telkom Purwokerto" menghasilkan output "otrekowruP mokleT".

**Untuk Codingannya lebih lengkapnya dibawah ini :**

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

class Stack {
private:
    int top;
    string arr[100];

public:

    Stack() {
        top = -1;
    }

    // Fungsi untuk menambah data ke stack
    void push(string data) {
        arr[++top] = data;
    }

    // Fungsi untuk memeriksa apakah stack kosong
```

```

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

// Fungsi untuk mengambil data dari stack
string pop() {
    if (!isEmpty())
        return arr[top--];
    return "";
}
};

int main() {
    Stack s;
    string input;
    string kata = "";

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

    for (char c : input) {
        if (c == ' ') {
            if (kata != "") {
                s.push(kata);
                kata = "";
            }
        } else {
            kata = c + kata;
        }
    }

    if (kata != "") {
        s.push(kata);
    }

    // Menampilkan stack
    cout << "Datastack Array : " << endl;
    cout << "Data : ";

    while (!s.isEmpty()) {
        cout << s.pop() << " ";
    }

    cout << endl;

    return 0;
}

```

## V. Kesimpulan

Stack adalah struktur data yang menerapkan prinsip Last In, First Out (LIFO), di mana elemen yang terakhir dimasukkan adalah yang pertama dikeluarkan. Dengan operasi dasar seperti push, pop, dan peek, stack memungkinkan pengelolaan data yang efisien dan terstruktur. Stack memiliki berbagai aplikasi penting dalam pemrograman, seperti manajemen memori, evaluasi ekspresi, dan pelacakan panggilan fungsi. Struktur data ini sangat berguna dalam masalah yang membutuhkan pengelolaan urutan atau data sementara, menjadikannya komponen penting dalam pengembangan perangkat lunak.











