

LAPORAN PRAKTIKUM

Modul 7

“STACK”



Disusun Oleh:

Nama : Ganes Gemi Putra

NIM : 2311104075

Kelas : SE-07-02

Dosen : WAHYU ANDI SAPUTRA

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. Tujuan

Mampu memahami konsep stack pada struktur data dan algoritma
Mampu mengimplementasikan operasi-operasi pada stack
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.

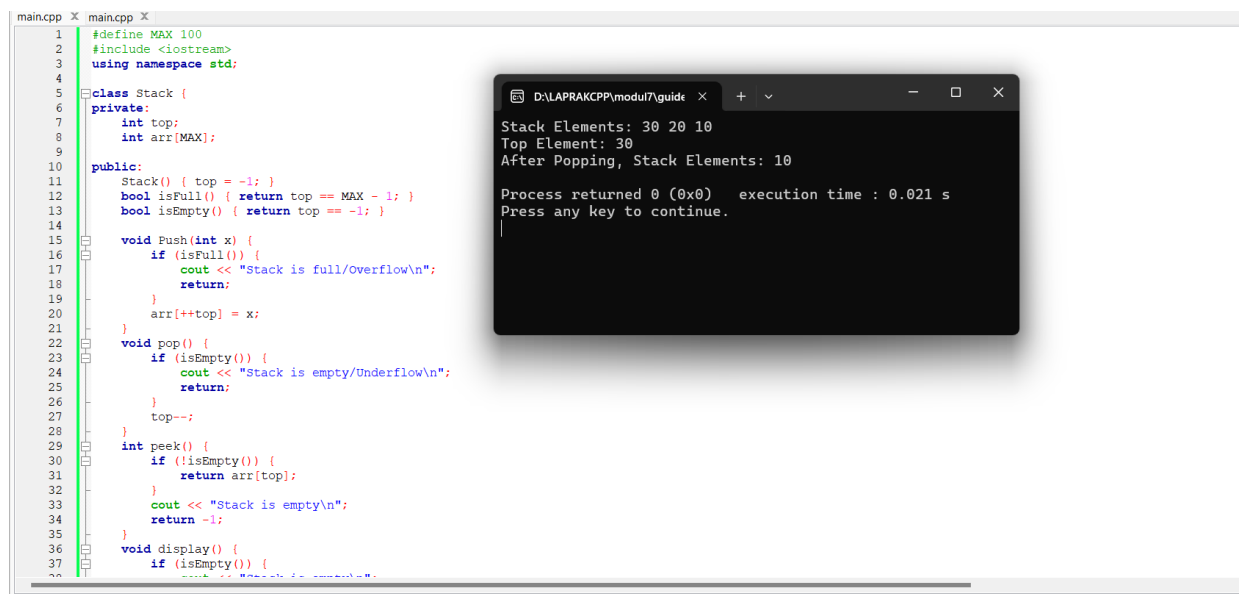
Dalam implementasinya, stack dapat direpresentasikan sebagai struktur data terurut yang memiliki dua operasi utama: push dan pop. Operasi push digunakan untuk menambahkan elemen baru ke dalam stack, sementara operasi pop digunakan untuk menghapus elemen teratas dari stack.

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..

3. Guided

1



The image shows a C++ program in a code editor and its execution output in a terminal window. The code defines a Stack class with a fixed-size array and a top pointer. It includes methods for push, pop, peek, isFull, isEmpty, and display. The main function demonstrates these operations by pushing 30, 20, and 10, then popping them, and finally displaying the stack state.

```
main.cpp x main.cpp x
1 #define MAX 100
2 #include <iostream>
3 using namespace std;
4
5 class Stack {
6 private:
7     int top;
8     int arr[MAX];
9
10 public:
11     Stack() { top = -1; }
12     bool isFull() { return top == MAX - 1; }
13     bool isEmpty() { return top == -1; }
14
15     void Push(int x) {
16         if (isFull()) {
17             cout << "Stack is full/overflow\n";
18             return;
19         }
20         arr[++top] = x;
21     }
22     void pop() {
23         if (isEmpty()) {
24             cout << "Stack is empty/Underflow\n";
25             return;
26         }
27         top--;
28     }
29     int peek() {
30         if (!isEmpty()) {
31             return arr[top];
32         }
33         cout << "Stack is empty\n";
34         return -1;
35     }
36     void display() {
37         if (isEmpty()) {
38             cout << "Stack is empty\n";
39         }
40     }
41 }
```

```
D:\LAPRAKCPP\modul7\guide x + -
Stack Elements: 30 20 10
Top Element: 30
After Popping, Stack Elements: 10

Process returned 0 (0x0)   execution time : 0.021 s
Press any key to continue.
```

2

The screenshot shows a C++ IDE with a file named `main.cpp`. The code implements a stack using a linked list structure. It includes a `Node` class with `data` and `next` pointers, and a `Stack` class with `top` pointer and methods for `push`, `pop`, and `peek`. The `main` function is partially visible at the bottom.

```

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() {top = nullptr;};
20     bool isEmpty() {return top == nullptr;};
21
22     void push(int x) {
23         Node* newNode = new Node(x);
24         newNode->next = top;
25         top = newNode;
26     }
27
28     void pop() {
29         if(isEmpty()) {
30             cout << "Stack is Underflow/empty\n";
31             return;
32         }
33         Node* temp = top;
34         top = top->next;
35         delete temp;
36     }
37
38     int peek() {
39         if(!isEmpty()) {
40             return top->data;
41         }
42     }
43 };
44
45 int main() {
46     Stack s;
47     s.push(10);
48     s.push(20);
49     s.push(30);
50     cout << "Stack Elements: 30 20 10\n";
51     cout << "Top Element: 30\n";
52     s.pop();
53     cout << "After Popping, Stack Elements: 10\n";
54     return 0;
55 }

```

The output window shows the following text:

```

D:\LAPRAKCPP\modul7\guide x + -
Stack Elements: 30 20 10
Top Element: 30
After Popping, Stack Elements: 10

Process returned 0 (0x0)   execution time : 0.027 s
Press any key to continue.

```

4. Unguided

The screenshot shows a C++ IDE with a file named `main.cpp`. The code implements a function `isPalindromeWithStack` that checks if a string is a palindrome using a stack. It includes a `Stack` class and a `main` function that takes a string input and checks if it is a palindrome.

```

1 #include <iostream>
2 #include <stack>
3 #include <algorithm>
4 #include <cctype>
5 using namespace std;
6
7 // Fungsi untuk memeriksa apakah kalimat adalah palindrome
8 bool isPalindromeWithStack(string sentence) {
9     // Normalisasi kalimat: hilangkan spasi dan ubah ke huruf kecil
10    string normalized = "";
11    for (char ch : sentence) {
12        if (isalnum(ch)) { // Memeriksa apakah karakter adalah alphanumeric
13            normalized += tolower(ch);
14        }
15    }
16
17    // Inisialisasi stack
18    stack<char> charStack;
19
20    // Masukkan setiap karakter ke dalam stack
21    for (char ch : normalized) {
22        charStack.push(ch);
23    }
24
25    // Bangun ulang kalimat dari stack (pop elemen satu per satu)
26    string reversed = "";
27    while (!charStack.empty()) {
28        reversed += charStack.top();
29        charStack.pop();
30    }
31
32    // Periksa apakah kalimat asli sama dengan kalimat terbalik
33    return normalized == reversed;
34 }
35
36 int main() {
37     string sentence;
38     cout << "Masukkan kalimat: ";
39     getline(cin, sentence);
40     if (isPalindromeWithStack(sentence)) {
41         cout << "Kalimat tersebut adalah: Palindrom\n";
42     } else {
43         cout << "Kalimat tersebut adalah: Bukan Palindrom\n";
44     }
45     return 0;
46 }

```

The output window shows the following text:

```

D:\LAPRAKCPP\modul7\ungu x + -
Masukkan kalimat: telkom
Kalimat tersebut adalah: Bukan Palindrom

Process returned 0 (0x0)   execution time : 26.699 s
Press any key to continue.

```

5. Kesimpulan

1. Manajemen memori komputer: Stack digunakan untuk melacak panggilan fungsi atau variabel lokal.
2. Evaluasi ekspresi aritmatika: Digunakan untuk menyimpan operator dan operand sementara.
3. Pembalikan string atau kalimat: Memanfaatkan prinsip LIFO untuk membalik elemen.

Program yang diberikan juga menunjukkan implementasi stack menggunakan:

- Array statis: Dengan batasan ukuran tertentu.

- **Linked list: Dengan fleksibilitas ukuran dinamis.**