

LAPORAN PRAKTIKUM

Modul 7

“STACK



Disusun Oleh:

Rengganis Tantri Pramudita - 2311104065

S1SE0702

Dosen :

Wahyu Andy Saputra

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

1. Tujuan

- Mampu memahami konsep stack pada struktur data dan algoritma
- Mampu mengimplementasikan operasi-operasi pada stack
- Mampu memecahkan permasalahan dengan solusi stack

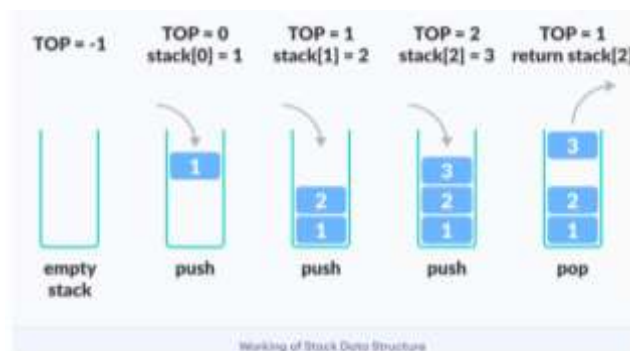
2. Landasan Teori

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..



Operasi pada stack melibatkan beberapa fungsi dasar yang dapat dilakukan pada struktur data ini. Berikut adalah beberapa operasi umum pada stack:

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

3. Guided

Guided 1

```

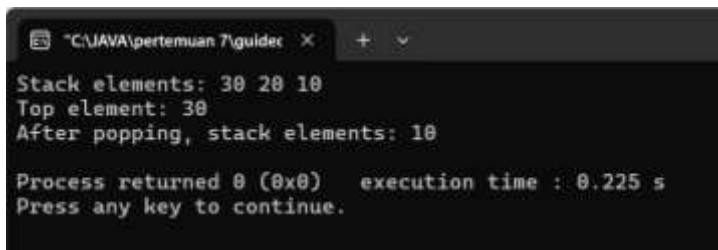
1 //Stack Implementation
2 #include <iostream>
3 #include <bits/stdc++.h>
4
5 using namespace std;
6
7 class Stack {
8     private:
9         int top;
10        int arr[100];
11
12    public:
13        Stack() { top = -1; }
14
15        bool isEmpty() { return top == -1; }
16        bool isFull() { return top == 99; }
17
18        void push(int x) {
19            if (isFull()) {
20                cout << "Stack Overflow!\n";
21                return;
22            }
23            arr[++top] = x;
24        }
25
26        void pop() {
27            if (isEmpty()) {
28                cout << "Stack Underflow!\n";
29                return;
30            }
31            top--;
32        }
33
34        int peek() {
35            if (isEmpty()) {
36                return arr[top];
37            }
38            cout << "Stack is empty!\n";
39            return -1; // Return a sentinel value
40        }
41
42        void display() {
43            if (isEmpty()) {
44                cout << "Stack is empty!\n";
45                return;
46            }
47            for (int i = top; i >= 0; i--) {
48                cout << arr[i] << " ";
49            }
50            cout << "\n";
51        }
52
53        int main() {
54            Stack s;
55            s.push(10);
56            s.push(20);
57            s.push(30);
58
59            cout << "Stack elements: ";
60            s.display();
61
62            cout << "Top element: " << s.peek() << "\n";
63
64            s.pop();
65            s.pop();
66            cout << "After popping, stack elements: ";
67            s.display();
68            return 0;
69        }
70    }

```

Keterangan

Program ini mengimplementasikan struktur data Stack (tumpukan) menggunakan array dengan ukuran maksimal 100 elemen. Stack direpresentasikan dalam sebuah class yang memiliki dua data member private yaitu variabel 'top' untuk menandai posisi elemen teratas dan array 'arr' untuk menyimpan elemen-elemen stack. Class ini dilengkapi dengan berbagai metode: konstruktor untuk inialisasi top = -1 (stack kosong), isFull() untuk mengecek apakah stack penuh, isEmpty() untuk mengecek apakah stack kosong, push() untuk menambah elemen baru ke stack, pop() untuk menghapus elemen teratas, peek() untuk melihat elemen teratas tanpa menghapusnya, dan display() untuk menampilkan semua elemen dalam stack. Dalam fungsi main(), program membuat objek Stack 's' kemudian melakukan serangkaian operasi: push(10), push(20), push(30) untuk menambahkan angka-angka tersebut ke dalam stack, menampilkan isi stack, melihat elemen teratas menggunakan peek(), melakukan dua kali operasi pop() untuk menghapus dua elemen teratas, dan akhirnya menampilkan kembali isi stack yang tersisa. Program ini menerapkan prinsip LIFO (Last In First Out) dimana data yang terakhir dimasukkan akan menjadi data pertama yang dikeluarkan, dengan pembatasan ukuran maksimal stack menggunakan array statis.

Outputnya:



Guided 2



```

41 int isEmpty() {
42     if (isEmpty()) {
43         return true;
44     }
45     return false;
46 }
47
48 void display() {
49     if (isEmpty()) {
50         return;
51     }
52     Node* current = top;
53     while (current != NULL) {
54         cout << current->data << " ";
55         current = current->next;
56     }
57     cout << endl;
58 }
59
60 int main() {
61     Stack s;
62     s.push(10);
63     s.push(20);
64     s.push(30);
65
66     cout << "Stack elements: ";
67     s.display();
68     cout << "Top element: " << s.peek() << endl;
69     s.pop();
70     cout << "After popping, stack elements: ";
71     s.display();
72     return 0;
73 }

```

Keterangan:

Stack ini diimplementasikan menggunakan struktur data Linked List (daftar berantai) dimana setiap elemen (node) memiliki dua bagian: data dan pointer ke node berikutnya. Program memiliki dua class utama yaitu Node untuk struktur data dan Stack untuk operasinya. Class Stack memiliki pointer 'top' yang selalu menunjuk ke elemen teratas. Operasi utamanya meliputi push (menambah elemen baru di depan), pop (menghapus elemen teratas), peek (melihat elemen teratas) dan display (menampilkan semua elemen). Berbeda dengan implementasi Stack menggunakan array, versi Linked List ini tidak memiliki batasan ukuran karena memory dialokasikan secara dinamis menggunakan operator 'new'. Dalam contoh di main(), program membuat Stack kosong lalu melakukan push(10), push(20), push(30) sehingga menghasilkan tumpukan 30 20 10, kemudian melakukan pop() untuk menghapus 30 sehingga tersisa 20 10. Cara kerja program tetap mengikuti prinsip LIFO (Last In First Out) dimana elemen yang terakhir masuk akan menjadi elemen pertama yang keluar.

Outputnya

```

C:\JAVA\pertemuan 7\guides x + ~
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 20 10

Process returned 0 (0x0)   execution time : 0.328 s
Press any key to continue.

```

4. Unguided

Soal no 1

- Code

```

1 // Fungsi isPalindrome
2 // Fungsi utama
3 // Fungsi isPalindrome
4 // Fungsi main
5
6 bool isPalindrome(string str) {
7     stack<char> s;
8     string strClean;
9
10    // Membersihkan string dari spasi
11    for (char c : str) {
12        if (c != ' ') {
13            strClean += c;
14        }
15    }
16
17    int length = strClean.length();
18    int mid = length / 2;
19
20    // Memeriksa apakah karakter pertama dan terakhir cocok
21    for (int i = 0; i < mid; i++) {
22        if (strClean[i] != strClean[length - i - 1]) {
23            return false;
24        }
25    }
26
27    // Jika panjang string ganjil, maka periksa tengah
28    if (length % 2 != 0) {
29        mid++;
30    }
31
32    // Memeriksa apakah karakter tengah cocok dengan tengah
33    for (int i = mid; i < length; i++) {
34        if (strClean[i] != strClean[length - i]) {
35            return false;
36        }
37    }
38
39    return true;
40 }
41
42 int main() {
43     string input;
44     cout << "Masukan Kalimat : ";
45     getline(cin, input);
46
47     cout << "Kalimat tersebut adalah : ";
48     if (isPalindrome(input)) {
49         cout << "Palindrom" << endl;
50     } else {
51         cout << "Bukan Palindrom" << endl;
52     }
53     return 0;
54 }
55

```

Keterangan

Program ini akan mendeteksi apakah sebuah kalimat adalah palindrom dengan mengabaikan spasi dan huruf besar/kecil.

1. Program menggunakan stack untuk menyimpan karakter-karakter dari string input.
 2. Langkah-langkah pemeriksaan palindrom:
 - Membersihkan string dari spasi dan mengubah ke huruf kecil
 - Memasukkan setengah pertama karakter ke dalam stack
 - Membandingkan setengah terakhir karakter dengan isi stack
 - Jika semua karakter cocok, maka string adalah palindrom
- Outputnya

```

Masukan Kalimat : ini
Kalimat tersebut adalah : Palindrom

```

```

Masukan Kalimat : Telkom
Kalimat tersebut adalah : Bukan Palindrom

```

Soal no 2

- Code

```

1 // Struktur Stack
2 #include <iostream>
3 #include <string>
4 using namespace std;
5
6 class Stack {
7 private:
8     int top;
9     string arr[100]; // Array untuk menyimpan kata
10
11 public:
12     Stack() {
13         top = -1;
14     }
15
16     void push(string data) {
17         arr[++top] = data;
18     }
19
20     bool isEmpty() {
21         return top == -1;
22     }
23
24     string pop() {
25         if (!isEmpty())
26             return arr[top--];
27         return "";
28     }
29 }
30
31 int main() {
32     Stack s;
33     string input;
34     string word = "";
35
36     cout << "Masukkan Kata : ";
37     getline(cin, input);
38
39     // Memeriksa kata apa saja yg ada
40     for(char c : input) {
41         if(c == ' ') {
42             if(word != "") {
43                 s.push(word);
44                 word = "";
45             }
46             else {
47                 word = c + word; // Menambahkan karakter ke kata
48             }
49         }
50         // kata telah selesai
51         if(word != "") {
52             s.push(word);
53         }
54
55         cout << "DataStack Array : ";
56         cout << "Data : ";
57
58         // Menampilkan hasil
59         while(!s.isEmpty()) {
60             cout << s.pop() << " ";
61         }
62         cout << endl;
63     }
64     return 0;
65 }

```

Keterangan

1. Struktur Program:

- Menggunakan class Stack untuk implementasi stack
- Method: push(), pop(), isEmpty()
- Array string untuk menyimpan kata-kata

2. Proses Pembalikan:

- Karakter dalam kata dibalik saat pembacaan input
- Kata yang sudah dibalik di-push ke stack
- Pop kata dari stack untuk mendapatkan urutan terbalik

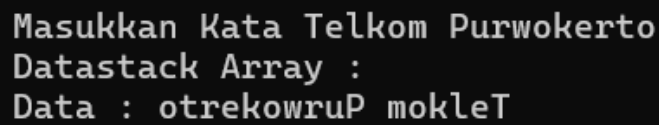
3. Fungsi-fungsi Stack:

- push(): Menambah kata ke stack
- pop(): Mengambil dan menghapus kata dari stack
- isEmpty(): Mengecek stack kosong

4. Alur Program:

- Input kalimat
- Balik setiap karakter dalam kata
- Push kata ke stack
- Pop dan tampilkan hasil

- Outputnya



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

5. Kesimpulan

Stack merupakan struktur data linier yang menggunakan prinsip LIFO (Last In First Out), dimana elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan. Dalam implementasinya menggunakan C++, stack dapat direalisasikan menggunakan array atau linked list dengan operasi dasar seperti push (menambah elemen), pop (menghapus elemen), peek/top (melihat elemen teratas), dan isEmpty (mengecek apakah stack kosong). Stack sangat berguna untuk menyelesaikan berbagai permasalahan komputasi seperti evaluasi ekspresi matematika, pencocokan tanda kurung, mengkonversi notasi infix ke postfix, serta tracking history pada browser. Penggunaan stack memudahkan pengelolaan data yang membutuhkan akses secara berurutan dari atas ke bawah dengan kompleksitas waktu $O(1)$ untuk operasi push dan pop.