

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL VII
STACK**



Disusun Oleh :

NAMA : Azzahra Fareluka Esti Ning Tyas

NIM : 103112430023

Dosen

FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

A. Dasar Teori

Stack adalah salah satu struktur data linear yang menggunakan prinsip Last In, First Out, yaitu elemen yang terakhir dimasukkan menjadi elemen pertama yang dikeluarkan. Konsep ini mirip dengan tumpukan buku, di mana buku yang paling atas adalah yang pertama kali dapat diambil. Dalam implementasinya, stack dapat dibuat dengan array statis ataupun linked list dinamis. Pada implementasi berbasis array, kapasitas stack sudah ditentukan sehingga jumlah elemen dibatasi oleh ukuran array. Operasi dasar seperti push dan pop memanipulasi indeks top untuk menambah atau menghapus elemen. Sedangkan pada implementasi berbasis linked list, stack dapat bertambah dan berkurang secara dinamis sesuai kebutuhan tanpa batasan ukuran tertentu karena setiap elemen diwakili oleh node yang terhubung melalui pointer.

Operasi fundamental dalam stack meliputi: push untuk menambahkan elemen ke bagian atas stack, pop untuk menghapus dan mengambil elemen teratas, peek/top untuk melihat nilai di puncak stack tanpa menghapusnya, serta isEmpty untuk memeriksa apakah stack kosong. Struktur data stack banyak digunakan dalam berbagai aplikasi komputer, seperti undo-redo, memory management, backtracking, hingga evaluasi ekspresi matematika. Dengan memahami cara kerja stack baik pada array maupun linked list, mahasiswa dapat mempelajari bagaimana sistem memanfaatkan struktur data ini dalam berbagai algoritma dan proses komputasi.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

```
C++ stack.cpp X
Modul07 > GUIDED > C++ stack.cpp > ...
1  #include <iostream>
2  using namespace std;
3
4  struct Node
5  {
6      int data;
7      Node* next;
8  };
9
10 bool isEmpty(Node* top)
11 {
12     return top == nullptr;
13 }
14
15 void push(Node *top, int data)
16 {
17     Node* newNode = new Node();
18     newNode->data = data;
19     newNode->next = top;
20     top = newNode;
21 }
22
23 int pop(Node *top)
24 {
25     if (isEmpty(top))
26     {
27         cout << "Stack kosong, tidak bisa pop!" << endl;
28         return 0;
29     }
30
31     int poppedData = top->data;
32     Node* temp = top;
33     top = top->next;
34
35     delete temp;
36     return poppedData;
37 }
38
39 void show(Node* top)
40 {
41     if (isEmpty(top))
42     {
43         cout << "Stack kosong." << endl;
44         return;
45     }
46
47     cout << "TOP -> ";
48     Node *temp = top;
49
50     while (temp != nullptr)
51     {
52         cout << temp->data << " -> ";
53         temp = temp->next;
54     }
55 }
56
57 int main()
58 {
59     Node *stack = nullptr;
60
61     push(stack, 10);
62     push(stack, 20);
63     push(stack, 30);
64
65     cout << "Menampilkan isi stack: " << endl;
66     show(stack);
67
68     cout << "Pop: " << pop(stack) << endl;
69
70     cout << "Menampilkan sisa stack: " << endl;
71     show(stack);
72
73     return 0;
74 }
```

Screenshots Output


```
PS D:\StrukturData\Modul07\UNGUIDED> cd "d:\StrukturData\Modul07\GUIDED\" ; if ($?) { g++ stack.cpp -o stack } ; if ($?) { .\stack }
Menampilkan isi stack:
TOP -> 30 -> 20 -> 10 -> Pop: 30
Menampilkan sisa stack:
TOP -> 20 -> 10 ->
```

Deskripsi:

Kode C++ tersebut adalah implementasi stack menggunakan linked list. Setiap elemen stack disimpan dalam struktur Nodee yang berisi data dan pointer ke node berikutnya. Stack dikelola menggunakan pointer top yang selalu menunjuk ke elemen teratas. Fungsi push menambah elemen baru ke bagian atas stack dengan membuat node baru dan menghubungkannya ke node sebelumnya. Fungsi pop menghapus node teratas, mengembalikan nilainya, dan menggeser pointer top ke node berikutnya. Fungsi isEmpty memeriksa apakah stack kosong, sedangkan show menampilkan seluruh isi stack dari atas ke bawah. Di fungsi main, beberapa data didorong ke stack, ditampilkan, lalu satu elemen di pop, dan sisa stack ditampilkan kembali. Program ini menunjukkan cara kerja stack secara dinamis menggunakan linked list tanpa batas ukuran tetap.

C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1



```
h stack.h X
Modul07 > UNGUIDED > h stack.h > ...
1  #ifndef STACK_H_INCLUDED
2  #define STACK_H_INCLUDED
3
4  #include <iostream>
5  using namespace std;
6
7  typedef int infotype;
8
9  struct Stack {
10     infotype info[20];
11     int top;
12 };
13
14 void createStack(Stack &S);
15 void push(Stack &S, infotype x);
16 infotype pop(Stack &S);
17 void printInfo(Stack S);
18 void balikStack(Stack &S);
19
20 #endif
```

```

C++ stack.cpp X
Modul07 > UNGUIDED > C++ stack.cpp > ...
1  #include "stack.h"
2
3  Tabnine | Edit | Test | Explain | Document
4  void createStack(Stack &S) {
5      S.top = -1;
6  }
7
8  Tabnine | Edit | Test | Explain | Document
9  void push(Stack &S, infotype x) {
10     if (S.top < 19) {
11         S.top++;
12         S.info[S.top] = x;
13     } else {
14         cout << "Stack penuh!" << endl;
15     }
16 }
17
18 Tabnine | Edit | Test | Explain | Document
19 infotype pop(Stack &S) {
20     if (S.top >= 0) {
21         infotype x = S.info[S.top];
22         S.top--;
23         return x;
24     } else {
25         cout << "Stack kosong!" << endl;
26         return -1;
27     }
28 }
29
30 Tabnine | Edit | Test | Explain | Document
31 void printInfo(Stack S) {
32     cout << "[TOP] ";
33     for (int i = S.top; i >= 0; i--) {
34         cout << S.info[i] << " ";
35     }
36     cout << endl;
37 }
38
39 Tabnine | Edit | Test | Explain | Document
40 void balikStack(Stack &S) {
41     Stack temp;
42     createStack(temp);
43
44     while (S.top >= 0) {
45         push(temp, pop(S));
46     }
47
48     S = temp;
49 }
50

```

```
C++ main.cpp X
Modul07 > UNGUIDED > C++ main.cpp > ...
1  #include <iostream>
2  #include "stack.h"
3  #include "stack.cpp"
4  using namespace std;
5
6  Tabnine | Edit | Test | Explain | Document
7  int main() {
8      cout << "Hello world!" << endl;
9
10     Stack S;
11     createStack(S);
12
13     push(S, 3);
14     push(S, 4);
15     push(S, 8);
16     pop(S);
17     push(S, 2);
18     push(S, 3);
19     pop(S);
20     push(S, 9);
21
22     printInfo(S);
23
24     cout << "balik stack" << endl;
25     balikStack(S);
26
27     printInfo(S);
28
29     return 0;
30 }
```

Screenshots Output

```
PS D:\StrukturData> cd "d:\StrukturData\Modul07\UNGUIDED\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
Hello world!
[TOP] 9 2 4 3
balik stack
[TOP] 3 4 2 9
```

Deskripsi:

Program di atas merupakan implementasi struktur data stack menggunakan array statis dengan kapasitas 20 elemen. Stack direpresentasikan oleh struct yang memiliki sebuah array info untuk menyimpan data dan variabel top sebagai penanda posisi elemen teratas. Fungsi createStack menginisialisasi stack agar kosong dengan mengatur top = -1. Operasi dasar stack juga diimplementasikan, yaitu push untuk menambah elemen ke puncak stack selama belum penuh, dan pop untuk mengambil serta menghapus elemen dari puncak stack selama tidak kosong. Fungsi printInfo digunakan untuk menampilkan seluruh isi stack mulai dari elemen teratas hingga terbawah. Selain itu, terdapat fungsi balikStack yang membalik urutan data dalam stack dengan memanfaatkan stack sementara: semua elemen dipindahkan dengan cara dipop dari stack asli lalu dipush ke stack sementara sehingga urutannya terbalik, kemudian hasilnya dikembalikan ke stack awal. Pada fungsi main, program melakukan serangkaian operasi push dan pop, menampilkan isi stack, lalu

membalik stack dan menampilkan hasilnya kembali. Program ini menunjukkan konsep dasar stack LIFO (Last In, First Out) serta cara membalik isi stack menggunakan operasi stack itu sendiri.

D. Kesimpulan

Berdasarkan praktikum yang dilakukan, dapat disimpulkan bahwa struktur data stack merupakan salah satu struktur data penting yang bekerja dengan prinsip LIFO. Melalui dua implementasi yang telah dipraktikkan menggunakan array statis dan menggunakan linked list dapat dilihat bahwa setiap pendekatan memiliki karakteristik masing-masing. Implementasi array lebih sederhana namun memiliki batas kapasitas tetap, sedangkan implementasi linked list lebih fleksibel karena ukurannya dapat berkembang dinamis sesuai kebutuhan. Operasi dasar seperti push, pop, dan pengecekan kondisi stack dapat dilakukan dengan mudah pada kedua metode tersebut. Selain itu, praktikum juga menunjukkan bagaimana stack dapat dimanipulasi lebih lanjut, seperti proses membalik isi stack menggunakan struktur stack itu sendiri. Dengan memahami konsep dan implementasinya, mahasiswa dapat menerapkan stack pada berbagai permasalahan komputasi secara efektif.

E. Referensi

<https://www.geeksforgeeks.org/dsa/stack-data-structure/>

<https://www.programiz.com/dsa/stack>

<https://en.cppreference.com/w/cpp/container/list.html>

https://www.tutorialspoint.com/data_structures_algorithms/stack_algorithm.htm