

**LAPORAN PRAKTIKUM
STRUKTUR DATA
MODUL 6**

(Dosen Pengampu : Toni Khalimi S.SI.,M.Kom.)



Disusun oleh : Arie Muhamad Syahrial
(20240810091) TINFC-2024-02

**PRODI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER UNIVERSITAS
KUNINGAN
2025**

D. PRE-TEST

1. **Apa yang dimaksud dengan struktur data Stack dan bagaimana prinsip kerjanya?**
Stack adalah struktur data linear yang mengikuti prinsip *LIFO* (*Last-In-First-Out*), artinya elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan. Operasi hanya dilakukan pada satu sisi, yaitu bagian atas (top) stack.
2. **Sebutkan dan jelaskan empat operasi dasar yang dapat dilakukan pada struktur data Stack!**
 - o **Push:** Menambahkan elemen ke atas stack.
 - o **Pop:** Menghapus elemen teratas dari stack dan mengembalikannya.
 - o **Peek/Top:** Mengembalikan nilai elemen teratas tanpa menghapusnya.
 - o **isEmpty:** Mengecek apakah stack kosong.
3. **Dalam bahasa pemrograman C atau C++, bagaimana cara mendeteksi apakah stack dalam keadaan kosong atau penuh ketika menggunakan array?**
 - o **Stack kosong:** `top == -1`
 - o **Stack penuh:** `top == MAX_SIZE - 1`
Di mana `MAX_SIZE` adalah batas maksimum kapasitas stack yang ditentukan.

PRAKTIKUM

1.

```
⌚ m6-1.cpp > ⚡ main()
1 #include <stdio.h>
2 #include <string.h>
3
4 Qodo Gen: Options | Test this function
5 void reverseString(char *str) {
6     int length = strlen(str);
7     for (int i = 0; i < length / 2; i++) {
8         char temp = str[i];
9         str[i] = str[length - i - 1];
10        str[length - i - 1] = temp;
11    }
12 }
13 Qodo Gen: Options | Test this function
14 int main() {
15     char str[] = "INDONESIA RAYA MERDEKA";
16     printf("Sebelum dibalik: %s\n", str);
17
18     reverseString(str);
19     printf("Setelah dibalik: %s\n", str);
20
21     return 0;
22 }
```

ANALISA PROGRAM :

◊ `#include <stdio.h>`

- **Fungsi:** Direktif preprocessor untuk memasukkan library standar `stdio.h` (Standard Input Output).
- **Tujuan:** Agar bisa menggunakan fungsi seperti `printf()` dan `scanf()`.

◊ `#include <string.h>`

- **Fungsi:** Menyisipkan library `string.h` yang berisi fungsi-fungsi manipulasi string seperti `strlen()`.
 - **Tujuan:** Agar kita bisa menghitung panjang string menggunakan `strlen()`.
-

```
◊ void reverseString(char *str)
```

- **Penjelasan:**
 - void menandakan fungsi ini **tidak mengembalikan nilai**.
 - reverseString adalah nama fungsi.
 - char *str berarti parameter str adalah **pointer ke karakter** (string di C adalah array karakter).
 - **Fungsi ini akan membalik isi string secara langsung (in-place).**
-

```
◊ int length = strlen(str);
```

- **strlen(str)** menghitung jumlah karakter dalam string **tanpa menghitung \0 (null terminator)**.
 - Misalnya: "ABC" → panjangnya 3.
 - Nilai panjang disimpan dalam variabel length.
-

```
◊ for (int i = 0; i < length / 2; i++)
```

- **Tujuan:** Melakukan pertukaran karakter **dari dua sisi string secara bertahap ke tengah**.
 - Misalnya string "MERDEKA":
 - Panjang = 7 → loop berjalan dari i = 0 sampai i = 2.
 - Indeks yang ditukar:
 - i=0 dengan 6
 - i=1 dengan 5
 - i=2 dengan 4
-

```
◊ char temp = str[i];
```

- Simpan karakter awal (str[i]) ke dalam variabel sementara temp.
-

```
◊ str[i] = str[length - i - 1];
```

- Isi str[i] digantikan dengan karakter dari ujung (belakang).
 - Misalnya: i=0 → str[0] diisi str[length - 0 - 1] = str[length - 1]
-

```
◊ str[length - i - 1] = temp;
```

- Tempat karakter belakang diisi dengan karakter awal yang tadi disimpan di temp.
-

```
◊ int main()
```

- Fungsi utama dari program. Tempat di mana eksekusi program dimulai.
-

```
◊ char str[] = "INDONESIA RAYA MERDEKA";
```

- Mendeklarasikan dan menginisialisasi string.
 - `char str[]` otomatis mengalokasikan panjang array sesuai isi string (termasuk `\0` di akhir).
 - "INDONESIA RAYA MERDEKA" panjangnya 23 karakter (tanpa `\0`), total array = 24 slot.
-

```
◊ printf("Sebelum dibalik: %s\n", str);
```

- Menampilkan string awal sebelum dibalik.
 - `%s` digunakan untuk menampilkan string.
-

```
◊ reverseString(str);
```

- Memanggil fungsi `reverseString`, mengirimkan string sebagai parameter.
 - Karena parameter bertipe pointer, maka fungsi bisa **mengubah isi asli string** (bukan hanya salinan).
-

```
◊ printf("Setelah dibalik: %s\n", str);
```

- Menampilkan string setelah proses pembalikan dilakukan.
-

```
◊ return 0;
```

- Mengembalikan nilai 0 menandakan bahwa program telah selesai dengan sukses.
-

◀ END Output Program:

```
PS C:\Users\Hype\Documents\struktur data\modul6> cd  
Sebelum dibalik: INDONESIA RAYA MERDEKA  
Setelah dibalik: AKEDREM AYAR AISENODNI  
PS C:\Users\Hype\Documents\struktur data\modul6> █
```

❖ Kesimpulan:

- Program ini melakukan pembalikan string *in-place* menggunakan algoritma pertukaran dua sisi.

- Tidak menggunakan array tambahan, efisien memori.
- Praktik pointer dan manipulasi array di C yang sangat umum dan penting.

2.

```
C: m6-2.cpp > ⚙ display(Stack *)  
1  #include <stdio.h>  
2  #include <stdlib.h>  
3  #define MAX 10 // Ukuran maksimum stack  
4  
5  struct Stack {  
6      int top;  
7      int arr[MAX];  
8  };  
9  
10 Qodo Gen: Options | Test this function  
11 void initStack(struct Stack *s) {  
12     s->top = -1;  
13 }  
14 Qodo Gen: Options | Test this function  
15 int isEmpty(struct Stack *s) {  
16     return s->top == -1;  
17 }  
18 Qodo Gen: Options | Test this function  
19 int isFull(struct Stack *s) {  
20     return s->top == MAX - 1;  
21 }  
22 Qodo Gen: Options | Test this function  
23 void push(struct Stack *s, int value) {  
24     if (isFull(s)) {  
25         printf("Stack penuh! Tidak dapat menambahkan elemen.\n");  
26         return;  
27     }  
28     s->arr[++(s->top)] = value;  
29     printf("%d dimasukkan ke dalam stack.\n", value);  
30 }  
31 Qodo Gen: Options | Test this function  
32 int pop(struct Stack *s) {  
33     if (isEmpty(s)) {  
34         printf("Stack kosong! Tidak ada elemen untuk dihapus.\n");  
35         return -1;  
36     }  
37     return s->arr[(s->top)--];  
38 }  
39 Qodo Gen: Options | Test this function  
40 int peek(struct Stack *s) {  
41     if (isEmpty(s)) {  
42         printf("Stack kosong! Tidak ada elemen untuk dilihat.\n");  
43         return -1;  
44     }  
45     return s->arr[s->top];  
46 }  
47 Qodo Gen: Options | Test this function  
48 void display(struct Stack *s) {  
49     if (isEmpty(s)) [  
          printf("Stack kosong!\n");
```

```

47 void display(struct Stack *s) {
48     if (isEmpty(s)) {
49         printf("Stack kosong!\n");
50         return;
51     }
52     printf("Isi stack: ");
53     for (int i = 0; i <= s->top; i++) {
54         printf("%d ", s->arr[i]);
55     }
56     printf("\n");
57 }
58
Qodo Gen: Options | Test this function
59 int main() {
60     struct Stack s;
61     initStack(&s);
62     int choice, value;
63
64     while (1) {
65         printf("\nMenu:\n");
66         printf("1. Push\n");
67         printf("2. Pop\n");
68         printf("3. Peek\n");
69         printf("4. Display\n");
70         printf("5. Exit\n");
71         printf("Pilih operasi: ");
72         scanf("%d", &choice);
73
74         switch (choice) {
75             case 1:
76                 printf("Masukkan nilai: ");
77                 scanf("%d", &value);
78                 push(&s, value);
79                 break;
80             case 2:
81                 printf("Menghapus elemen: %d\n", pop(&s));
82                 break;
83             case 3:
84                 printf("Elemen teratas: %d\n", peek(&s));
85                 break;
86             case 4:
87                 display(&s);
88                 break;
89             case 5:
90                 exit(0);
91             default:
92                 printf("Pilihan tidak valid!\n");
93         }
94     }
95     return 0;
96 }
```

Bagian Preprocessor dan Konstanta

```
#include <stdio.h>
```

Digunakan untuk fungsi input-output standar seperti `printf` dan `scanf`.

```
#include <stdlib.h>
```

Digunakan untuk fungsi `exit()` yang menghentikan program.

```
#define MAX 10
```

Mendefinisikan konstanta `MAX` sebagai ukuran maksimum array dalam stack, yaitu 10 elemen.

Struktur Stack

```
struct Stack {  
    int top;  
    int arr[MAX];  
};
```

- `top`: indeks elemen teratas dalam stack.
 - `arr[MAX]`: array dengan panjang tetap sebagai tempat penyimpanan elemen stack.
-

Fungsi Inisialisasi Stack

```
void initStack(struct Stack *s) {  
    s->top = -1;  
}
```

Menginisialisasi `top` menjadi `-1`, yang menandakan stack masih kosong.

Fungsi Pengecekan Kosong dan Penuh

```
int isEmpty(struct Stack *s) {  
    return s->top == -1;  
}
```

Mengembalikan nilai 1 (true) jika `top == -1`, artinya stack kosong.

```
int isFull(struct Stack *s) {  
    return s->top == MAX - 1;  
}
```

Mengembalikan nilai 1 (true) jika `top` mencapai indeks maksimum (`MAX - 1`), artinya stack penuh.

Fungsi Push

```
void push(struct Stack *s, int value) {  
    if (isFull(s)) {  
        printf("Stack penuh! Tidak dapat menambahkan elemen.\n");  
        return;  
    }  
    s->arr[+(s->top)] = value;  
    printf("%d dimasukkan ke dalam stack.\n", value);  
}
```

- Jika stack penuh, akan menampilkan pesan dan keluar dari fungsi.
- Jika tidak, `top` dinaikkan (`++top`) kemudian nilai dimasukkan ke posisi tersebut.

Contoh: jika `top = -1`, maka `++top = 0`, nilai masuk ke `arr[0]`.

Fungsi Pop

```
int pop(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack kosong! Tidak ada elemen untuk dihapus.\n");
        return -1;
    }
    return s->arr[(s->top)--];
}
```

- Jika kosong, tampilkan pesan dan kembalikan -1.
 - Jika tidak, ambil nilai `arr[top]` lalu turunkan `top`.
-

Fungsi Peek

```
int peek(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack kosong! Tidak ada elemen untuk dilihat.\n");
        return -1;
    }
    return s->arr[s->top];
}
```

Mengembalikan nilai teratas dari stack tanpa menghapusnya.

Fungsi Display

```
void display(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack kosong!\n");
        return;
    }
    printf("Isi stack: ");
    for (int i = 0; i <= s->top; i++) {
        printf("%d ", s->arr[i]);
    }
    printf("\n");
}
```

Menampilkan seluruh elemen stack dari bawah ke atas (`arr[0]` ke `arr[top]`).

Fungsi Main

```
int main() {
    struct Stack s;
    initStack(&s);
    int choice, value;
```

- Membuat variabel `s` bertipe `struct Stack`.
 - Menginisialisasi stack menggunakan `initStack`.
 - Menyiapkan variabel `choice` untuk menu dan `value` untuk nilai input.
-

Menu Interaktif

```
while (1) {
    printf("\nMenu:\n");
    printf("1. Push\n");
    printf("2. Pop\n");
    printf("3. Peek\n");
    printf("4. Display\n");
    printf("5. Exit\n");
    printf("Pilih operasi: ");
    scanf("%d", &choice);
```

Menampilkan menu terus menerus menggunakan `while (1)`, dan membaca pilihan user melalui `scanf`.

Struktur Switch

```
switch (choice) {
    case 1:
        printf("Masukkan nilai: ");
        scanf("%d", &value);
        push(&s, value);
        break;
    case 2:
        printf("Menghapus elemen: %d\n", pop(&s));
        break;
    case 3:
        printf("Elemen teratas: %d\n", peek(&s));
        break;
    case 4:
        display(&s);
        break;
    case 5:
        exit(0);
    default:
        printf("Pilihan tidak valid!\n");
}
```

}

}

}

- `case 1:` memanggil `push()` untuk menambahkan elemen ke stack.
 - `case 2:` memanggil `pop()` dan mencetak nilai yang dihapus.
 - `case 3:` memanggil `peek()` dan mencetak nilai teratas stack.
 - `case 4:` menampilkan semua isi stack.
 - `case 5:` keluar dari program.
 - `default:` menangani input yang bukan dari 1–5.
-

Contoh Output (Simulasi Jalannya Program)

Misalnya input dari pengguna adalah:

```
1  
Masukkan nilai: 5  
1  
Masukkan nilai: 8  
3  
4  
2  
4  
5
```

Maka outputnya adalah:

```
5 dimasukkan ke dalam stack.  
8 dimasukkan ke dalam stack.  
Elemen teratas: 8  
Isi stack: 5 8  
Menghapus elemen: 8  
Isi stack: 5
```

Penjelasan:

- Nilai 5 dan 8 dimasukkan secara berurutan, stack sekarang: [5, 8].
- `peek()` mengembalikan 8 (teratas).
- `display()` menampilkan isi stack: 5 8.
- `pop()` menghapus 8, sekarang stack menjadi: [5].
- `display()` ulang menampilkan hanya 5.
- `exit(0)` menghentikan program.

```
Menu:  
1. Push  
2. Pop  
3. Peek  
4. Display  
5. Exit  
Pilih operasi: 2  
Menghapus elemen: 12  
  
Menu:  
1. Push  
2. Pop  
3. Peek  
4. Display  
5. Exit  
Pilih operasi: 1  
Masukkan nilai: 11  
11 dimasukkan ke dalam stack.  
  
Menu:  
1. Push  
2. Pop  
3. Peek  
4. Display  
5. Exit  
Pilih operasi: 1  
Masukkan nilai: 12  
12 dimasukkan ke dalam stack.  
  
Menu:  
1. Push  
2. Pop  
3. Peek  
4. Display  
5. Exit  
Pilih operasi: 3  
Elemen teratas: 12  
  
Menu:  
1. Push  
2. Pop  
3. Peek  
4. Display  
5. Exit  
Pilih operasi: 4  
Isi stack: 11 12  
  
Menu:  
1. Push  
2. Pop  
3. Peek  
4. Display  
5. Exit  
12 dimasukkan ke dalam stack.  
  
Menu:  
1. Push  
2. Pop  
3. Peek  
4. Display  
5. Exit  
Pilih operasi: 3  
Elemen teratas: 12
```

F. POST-TEST

1. **Sebutkan tiga contoh aplikasi nyata dari penggunaan Stack dalam dunia pemrograman atau sistem komputer!**
 - **Pemanggilan fungsi (Call Stack):** Saat fungsi dipanggil, alamat kembali dan variabel lokal disimpan di stack.
 - **Fitur Undo/Redo:** Digunakan di editor teks atau desain untuk menyimpan aksi pengguna.
 - **Evaluasi ekspresi matematika:** Seperti dalam notasi postfix (Reverse Polish Notation).
2. **Buatlah program C++ sederhana untuk mengimplementasikan struktur data Stack menggunakan array. Program harus memiliki operasi dasar seperti Push, Pop, dan Display. Sertakan pengecekan kondisi stack kosong dan penuh.**

```
4 posttest.cpp > ...
1 #include <iostream>
2 using namespace std;
3
4 #define MAX_SIZE 100
5
6 Qodo Gen: Options | Test this class
7 class Stack {
8 private:
9     int data[MAX_SIZE];
10    int top;
11
12 public:
13     Qodo Gen: Options | Test this method
14     Stack() {
15         top = -1;
16     }
17
18     Qodo Gen: Options | Test this method
19     void push(int value) {
20         if (top == MAX_SIZE - 1) {
21             cout << "Stack penuh (Overflow)!" << endl;
22         } else {
23             top++;
24             data[top] = value;
25             cout << "Elemen " << value << " berhasil ditambahkan ke stack." << endl;
26         }
27
28     Qodo Gen: Options | Test this method
29     void pop() {
30         if (top == -1) {
31             cout << "Stack kosong (Underflow)!" << endl;
32         } else {
33             cout << "Elemen " << data[top] << " dihapus dari stack." << endl;
34             top--;
35         }
36
37     Qodo Gen: Options | Test this method
38     void display() {
39         if (top == -1) {
40             cout << "Stack kosong." << endl;
41         } else {
42             cout << "Isi stack (dari atas ke bawah): ";
43             for (int i = top; i >= 0; i--) {
44                 cout << data[i] << " ";
45             }
46         }
47     }
48 }
```

```

40         for (int i = top; i >= 0; i--) {
41             cout << data[i] << " ";
42         }
43     cout << endl;
44 }
45 }
46 };
47
Qodo Gen: Options | Test this function
48 int main() {
49     Stack s;
50     int pilihan, nilai;
51
52     do {
53         cout << "\nMenu Stack:\n";
54         cout << "1. Push\n2. Pop\n3. Tampilkan Stack\n4. Keluar\n";
55         cout << "Pilih opsi: ";
56         cin >> pilihan;
57
58         switch (pilihan) {
59             case 1:
60                 cout << "Masukkan nilai: ";
61                 cin >> nilai;
62                 s.push(nilai);
63                 break;
64             case 2:
65                 s.pop();
66                 break;
67             case 3:
68                 s.display();
69                 break;
70             case 4:
71                 cout << "Program selesai.\n";
72                 break;
73             default:
74                 cout << "Pilihan tidak valid.\n";
75         }
76     } while (pilihan != 4);
77
78     return 0;
79 }
80

```

ANALISA:

1. Header dan Namespace

```
#include <iostream>
```

```
using namespace std;
```

- `#include <iostream>`: digunakan untuk input/output seperti `cin` dan `cout`.
 - `using namespace std;`: agar tidak perlu menuliskan `std::` sebelum `cout`, `cin`, dll.
-

2. Konstanta Ukuran Stack

```
#define MAX_SIZE 100
```

- Mendefinisikan ukuran maksimal stack sebagai 100 elemen.
 - Stack hanya bisa menampung maksimal `MAX_SIZE` data, indeks 0 sampai 99.
-

3. Deklarasi Kelas Stack

```
class Stack {  
private:  
    int data[MAX_SIZE]; // array untuk menyimpan data  
    int top; // penanda posisi elemen paling atas
```

- `data[MAX_SIZE]`: array sebagai struktur data stack.
 - `top`: menyimpan indeks elemen teratas. Jika `top == -1`, berarti stack kosong.
-

4. Konstruktor

```
public:  
    Stack() {  
        top = -1;  
    }
```

- Mengatur nilai awal `top` ke -1 saat objek stack dibuat.
 - Ini menandakan stack dalam kondisi kosong.
-

5. Fungsi `push(int value)`

```
void push(int value) {  
    if (top == MAX_SIZE - 1) {  
        cout << "Stack penuh (Overflow)!" << endl;  
    } else {  
        top++;  
        data[top] = value;  
        cout << "Elemen " << value << " berhasil ditambahkan ke stack." << endl;  
    }  
}
```

Logika:

- Jika `top == MAX_SIZE - 1`, berarti stack sudah penuh, maka tidak bisa menambahkan lagi (overflow).
- Jika belum penuh:
 - `top` dinaikkan (`++top`)
 - Nilai `value` dimasukkan ke `data[top]`.

Contoh:

Jika `top = 2`, maka:

- Setelah `push(10)`, `top` jadi 3
 - `data[3] = 10`
-

6. Fungsi `pop()`

```
void pop() {
    if (top == -1) {
        cout << "Stack kosong (Underflow)!" << endl;
    } else {
        cout << "Elemen " << data[top] << " dihapus dari stack." << endl;
        top--;
    }
}
```

Logika:

- Jika `top == -1`, stack kosong, maka tidak bisa menghapus (underflow).
 - Jika tidak kosong:
 - Cetak elemen yang akan dihapus (`data[top]`)
 - Turunkan `top` (`--top`), yang secara logika berarti elemen itu tak lagi dianggap bagian dari stack.
-

7. Fungsi `display()`

```
void display() {
    if (top == -1) {
        cout << "Stack kosong." << endl;
    } else {
        cout << "Isi stack (dari atas ke bawah): ";
        for (int i = top; i >= 0; i--) {
            cout << data[i] << " ";
        }
        cout << endl;
    }
}
```

Logika:

- Jika kosong, tampilkan pesan bahwa stack kosong.
 - Jika tidak:
 - Cetak elemen dari `top` ke 0, mencerminkan urutan stack (LIFO - Last In, First Out).
-

8. Fungsi `main()`

```
int main() {
    Stack s;
    int pilihan, nilai;
```

- Membuat objek stack `s`.
- `pilihan`: menyimpan pilihan menu dari user.

- nilai: menyimpan input angka dari user saat push.
-

Menu Interaktif (do-while loop)

```
do {  
    cout << "\nMenu Stack:\n";  
    cout << "1. Push\n2. Pop\n3. Tampilkan Stack\n4. Keluar\n";  
    cout << "Pilih opsi: ";  
    cin >> pilihan;  
  
    switch (pilihan) {  
        case 1:  
            cout << "Masukkan nilai: ";  
            cin >> nilai;  
            s.push(nilai);  
            break;  
        case 2:  
            s.pop();  
            break;  
        case 3:  
            s.display();  
            break;  
        case 4:  
            cout << "Program selesai.\n";  
            break;  
        default:  
            cout << "Pilihan tidak valid.\n";  
    }  
} while (pilihan != 4);
```

Penjelasan:

- Menampilkan menu hingga user memilih 4.
 - Pilihan:
 - 1: push nilai ke stack
 - 2: pop elemen dari stack
 - 3: tampilkan semua isi stack
 - 4: keluar dari program
 - Selain itu: tampilkan pesan bahwa pilihan tidak valid.
-

Simulasi Output (Contoh Jalannya Program)

Misalnya user melakukan input:

```
1  
Masukkan nilai: 15  
1  
Masukkan nilai: 27  
3  
2  
3  
4
```

Maka outputnya:

```
Elemen 15 berhasil ditambahkan ke stack.  
Elemen 27 berhasil ditambahkan ke stack.  
Isi stack (dari atas ke bawah): 27 15  
Elemen 27 dihapus dari stack.  
Isi stack (dari atas ke bawah): 15  
Program selesai.
```

Penjelasan:

- Pertama push(15), lalu push(27)
- Tampil isi stack: 27 (atas), 15 (bawah)
- pop() akan menghapus 27 (elemen paling atas)
- display() ulang menunjukkan sisa stack hanya 15
- Keluar saat user pilih 4