

# **LAPORAN AKHIR PRAKTIKUM ALGORITMA & STRUKTUR DATA**



**Oleh:**

**Indra Suryadilaga      NIM. 2410817310014**

**PROGRAM STUDI TEKNOLOGI INFORMASI  
FAKULTAS TEKNIK  
UNIVERSITAS LAMBUNG MANGKURAT  
2024**

**LEMBAR PENGESAHAN**  
**LAPORAN AKHIR PRAKTIKUM ALGORITMA & STRUKTUR**  
**DATA**

Laporan Akhir Praktikum Algoritma & Struktur Data

Modul 1: Struct & Pointer

Modul 2 :Stack & Queue

Modul 3: Single Link List

Modul 4: Double Link List

Modul 5: Sorting

Modul 6: Searching

Modul 7: Tree

ini disusun sebagai syarat lulus mata kuliah Praktikum Algoritma & Struktur Data.  
Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Indra Suryadilaga

NIM : 2410817310014

Menyetujui,  
Asisten Praktikum

Mengetahui,  
Dosen Penanggung Jawab Praktikum

Harry Pratama Yunus  
NIM. 2310817210010

Andreyan Rizky Baskara, S.Kom.,  
M.Kom  
NIP. 199307032019031011

## DAFTAR ISI

LEMBAR PENGESAHAN .....	2
DAFTAR ISI .....	3
DAFTAR GAMBAR .....	6
DAFTAR TABEL .....	8
MODUL 1 : STRUCT DAN POINTER .....	9
SOAL 1 .....	9
A. Source code .....	10
B. Output Program .....	10
C. Pembahasan .....	11
MODUL 2 : STACK DAN QUEUE.....	12
SOAL 1 .....	12
A. Pembahasan.....	12
SOAL 2 .....	13
A. Source Code .....	14
B. Output Program.....	18
C. Pembahasan.....	18
SOAL 3 .....	22
A. Source Code .....	23
B. Output Program.....	26
C. Pembahasan.....	26
MODUL 3 : SINGLE LINK LIST .....	30
SOAL 1 .....	30
A. Source Code .....	34
B. Output Program.....	45
SOAL 2 .....	46
A. Output Program.....	46
SOAL 3 .....	47
A. Output Program.....	47
SOAL 4 .....	48

A. Output Program.....	48
SOAL 5 .....	49
A. Output Program.....	49
SOAL 6 .....	50
A. Output Program.....	50
SOAL 7 .....	51
A. Output Program.....	51
SOAL 8 .....	52
A. Output Program.....	52
B. Pembahasan.....	52
SOAL 9 .....	53
A. Output Program.....	53
B. Pembahasan.....	53
SOAL 10 .....	54
A. Pembahasan.....	54
MODUL 4 : DOUBLE LINK LIST.....	55
SOAL 1 .....	55
A. Screenshoot .....	60
SOAL 2 .....	61
A. Pembahasan.....	61
SOAL 3 .....	62
A. Pembahasan.....	62
SOAL 4 .....	63
A. Pembahasan.....	63
MODUL 5 : SORTING.....	65
SOAL 1 .....	65
A. Source Code .....	66
A. Output Program.....	73
B. Pembahasan.....	74
MODUL 6 : SEARCHING .....	80
SOAL 1 .....	80

B. Source Code .....	82
C. Output Program.....	86
D. Pembahasan.....	88
MODUL 7 : TREE .....	92
SOAL 1 .....	92
B. Source Code .....	94
C. Output Program.....	99
D. Pembahasan.....	100
LINK GITHUB .....	104

## **DAFTAR GAMBAR**

### **MODUL 1 : STRUCT DAN POINTER**

Gambar 1. Modul 1 Soal 1 .....	9
Gambar 2. Modul 1 Output Saol 1 .....	10

### **MODUL 2 : STACK DAN QUEUE**

Gambar 3. Modul 2 soal 2.....	13
Gambar 4. Modul 2 Output Opsi Input Saol 2 .....	18
Gambar 5. Modul 2 Output Opsi Hapus Saol 2 .....	18
Gambar 6. Modul 2 Output Opsi Cetak Saol 2 .....	18
Gambar 7. Modul 2 Output Opsi Keluar Saol 2.....	18
Gambar 8. Modul 2 Soal 3 .....	22
Gambar 9. Modul 2 Output Opsi Insert Saol 3 .....	26
Gambar 10. Modul 2 Output Opsi Delete Saol 3 .....	26
Gambar 11. Modul 2 Output Opsi Cetak Saol 3 .....	26
Gambar 12. Modul 2 Output Opsi Quit Saol 3 .....	26

### **MODUL 3 : SINGLE LINK LIST**

Gambar 13. Modul 3 Soal 1 .....	33
Gambar 14. Modul 3 Output Saol 1 .....	45
Gambar 15. Modul 3 Screenshot Output Saol 2 .....	46
Gambar 16. Modul 3 Screenshot Output Saol 3 .....	47
Gambar 17. Modul 3 Screenshot Output Saol 4 .....	48
Gambar 18. Modul 3 Output Saol 5.....	49
Gambar 19. Modul 3 Output Saol 6.....	50
Gambar 20. Modul 3 Output Saol 7.....	51
Gambar 21. Modul 3 Output Saol 8.....	52
Gambar 22. Modul 3 Output Saol 9.....	53

## **MODUL 4 : DOUBLE LINK LIST**

Gambar 23. Modul 4 Saol 1 .....	59
Gambar 24. Modul 4 Output Saol 1 .....	60
Gambar 25. Modul 4. Output Perubahan baris 244 dan 256 Soal 4.....	63

## **MODUL 5 : SORTING**

Gambar 26. Modul 5 Soal 1 .....	65
Gambar 27. Modul 5 Output Insertion Sort Saol 1 .....	73
Gambar 28. Modul 5 Output Merge Sort Saol 1 .....	73
Gambar 29. Modul 5 Output Shell Sort Saol 1 .....	73
Gambar 30. Modul 5 Output Bubble Sort Saol 1 .....	73
Gambar 31. Modul 5 Output Quick Sort Saol 1 .....	73
Gambar 32. Modul 5 Output Selection Sort Saol 1 .....	73

## **MODUL 6 : SEARCHING**

Gambar 33. Modul 6 Sequential Searching Soal 1 .....	80
Gambar 34. Modul 6 Binary Searching Soal 1 .....	81
Gambar 35. Modul 6 Output Sequential Searching Saol 1 .....	86
Gambar 36. Modul 6 Output Binary Searching Saol 1 .....	87
Gambar 37. Modul 6 Output Perbedaan Sequential dan Binary Searching Saol 1 .....	87
Gambar 38. Modul 6 Soal 1 .....	94

## **MODUL 7 : TREE**

Gambar 39. Modul 7 Output Insert Saol 1 .....	99
Gambar 40. Modul 7 Output PreOrder Saol 1 .....	99
Gambar 41. Modul 7 Output InOrder Saol 1 .....	99
Gambar 42. Modul 7 Output PostOrder Saol 1 .....	100

## **DAFTAR TABEL**

### **MODUL 1 : STRUCT DAN POINTER**

Tabel 1. Modul 1 Source Code Soal 1 .....	10
Tabel 2. Modul 1 Source Code Soal 2 .....	14
Tabel 3. Modul 1 Source Code Soal 3 .....	23

### **MODUL 3 : SINGLE LINK LIST**

Tabel 4. Modul 3 Source Code Soal 1 .....	34
-------------------------------------------	----

### **MODUL 5 : SORTING**

Tabel 5. Modul 5 Source Code Soal 1 .....	66
-------------------------------------------	----

### **MODUL 6 : SEARCHING**

Tabel 6. Modul 6 Source Code Soal 1 .....	82
-------------------------------------------	----

### **MODUL 7 : TREE**

Tabel 7. Modul 7 Source Code Soal 1 .....	94
-------------------------------------------	----



# MODUL 1 : STRUCT DAN POINTER

## SOAL 1

Cobalah program berikut, running dan analisis hasilnya. Buatlah algoritma untuk program tersebut.

```
#include <iostream>

using namespace std;

struct mhs
{
    char nama[20], nim[10], jurusan[2];
    int sks, program;
};

struct mhs bayar[2];

main() {
    int bts, var, tetap;
    for(int i=0; i<2; i++)
    {
        //Input data
        cout<<"\n\n-----\n";
        cout<<"\nNama mhs          = ";cin>>bayar[i].nama;
        cout<<"\nNIM              = ";cin>>bayar[i].nim;
        cout<<"\nJurusan[TI, PTK]    = ";cin>>bayar[i].jurusan;
        input:
        cout<<"Program[1=D3, 2=S1] = ";
        cin>>bayar[i].program;

        if(bayar[i].program<0 || bayar[i].program>2)
        {
            cout<<"Program tidak sesuai\n";
            goto input;
        } cout<<"Jumlah sks          = "; cin>>bayar[i].sks;

        if(bayar[i].program==1)
        {
            tetap=500000;
            var=bayar[i].sks*25000;
        }else if(bayar[i].program==2)
        {
            tetap=750000;
            var=bayar[i].sks*50000;
        }cout<<"endl;

        //Output data
        cout<<"\n\n-----\n";
        cout<<" Output ";
        cout<<"\n\n-----\n";
        cout<<"\nNama mhs          = "<<bayar[i].nama;
        cout<<"\nNIM              = "<<bayar[i].nim;
        cout<<"\nJurusan          = "<<bayar[i].jurusan;
        cout<<"\nProgram          = "<<bayar[i].program;
        cout<<"\nJumlah sks        = "<<bayar[i].sks;
        cout<<"\nSPP tetap          = "<<tetap;
        cout<<"\nSPP variabel      = "<<var;
        cout<<"endl<<"endl;
    }
}
```

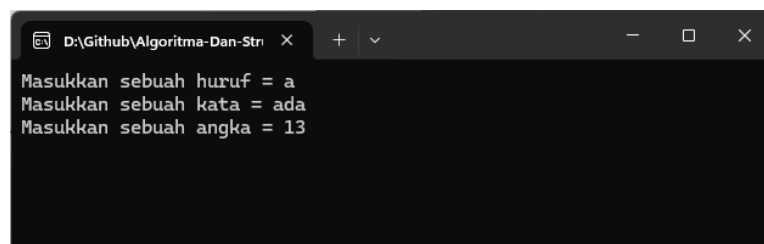
Gambar 1. Modul 1 Soal 1

## A. Source code

*Tabel 1. Modul 1 Source Code Soal 1*

```
1. #include <iostream>
2. using namespace std;
3.
4. int main() {
5.     char huruf;
6.     string kata;
7.     int angka;
8.
9.     // Input data
10.    cout << "Masukkan sebuah huruf = "; cin >> huruf;
11.    cin.ignore(); // Mengatasi masalah input buffer
13.    cout << "Masukkan sebuah kata = ";getline(cin, kata);
14.    cout << "Masukkan sebuah angka = ";cin >> angka;
15.
16.    // Output hasil
17.    cout << "\nHuruf yang Anda masukkan adalah " << huruf <<
18. endl;
19.    cout << "Kata yang Anda masukkan adalah " << kata << endl;
20.    cout << "Angka yang Anda masukkan adalah " << angka << endl;
21.    return 0;
22. }
```

## B. Output Program

A screenshot of a terminal window with a dark background. The window title bar shows the file path 'D:\Github\Algoritma-Dan-Str' and standard window controls. The terminal displays the program's output: 'Masukkan sebuah huruf = a', 'Masukkan sebuah kata = ada', and 'Masukkan sebuah angka = 13'. Each prompt is on a new line, and the user input is shown immediately after the prompt.

*Gambar 2. Modul 1 Output Saol 1*

### **C. Pembahasan**

Program pada gambar tersebut digunakan untuk menghitung biaya SPP berdasarkan program studi D3 atau S1 dan jumlah SKS yang diambil.

## **MODUL 2 : STACK DAN QUEUE**

### **SOAL 1**

Apa perbedaan Stack dengan Queue?

#### **A. Pembahasan**

Stack dan Queue merupakan dua jenis struktur data linear yang memiliki perbedaan utama dalam cara menyimpan dan mengambil data.

Stack bekerja dengan prinsip Last In, First Out (LIFO), artinya data yang terakhir dimasukkan akan menjadi data pertama yang dikeluarkan. Struktur ini dapat diibaratkan seperti tumpukan buku, di mana buku yang paling atas akan diambil terlebih dahulu. Operasi dasar pada Stack meliputi push (menambahkan data ke atas tumpukan) dan pop (menghapus data dari atas tumpukan).

Sebaliknya, Queue menggunakan prinsip First In, First Out (FIFO), yang berarti data yang pertama kali dimasukkan akan menjadi data pertama yang dikeluarkan. Queue dapat dianalogikan seperti antrean di loket, di mana orang yang datang lebih dahulu akan dilayani lebih dahulu. Operasi dasar pada Queue terdiri dari enqueue (menambahkan data ke belakang antrean) dan dequeue (menghapus data dari depan antrean).

Kedua struktur data ini sangat penting dalam berbagai aplikasi komputer, seperti manajemen memori, pemrosesan data, dan pengendalian antrian.

## SOAL 2

Cobalah contoh program berikut, running dan analisis hasilnya!

```
int penuh()
{
    if(Tumpuk.atas == max-1)
        return 1;
    else
        return 0;
}

void input (int data)
{
    if (kosong()==1)
    {
        Tumpuk.atas++;
        Tumpuk.data[Tumpuk.atas] = data;
        cout << "Data " << Tumpuk.data[Tumpuk.atas]
            << " Masuk Ke Stack ";
    }
    else if(penuh()==0)
    {
        Tumpuk.atas++;
        Tumpuk.data[Tumpuk.atas] = data;
        cout << "Data " << Tumpuk.data[Tumpuk.atas]
            << " Masuk Ke Stack ";
    }
    else
        cout << "Tumpukan Penuh";
}

void hapus()
{
    if(kosong()== 0)
    {
        cout << "Data Teratas Sudah Terambil";
        Tumpuk.atas--;
    }
    else
        cout << "Data Kosong";
}

void tampil()
{
    if (kosong()== 0)
    {
        for(int i = Tumpuk.atas; i>=0; i--)
        {
            cout << "\nTumpukan Ke " << i << " = "
                << Tumpuk.data[i];
        }
    }
    else
        cout << "Tumpukan Kosong";
}

void bersih ()
{
    Tumpuk.atas = -1;
    cout << "Tumpukan Kosong !";
}
```

Gambar 3. Modul 2 soal 2

## A. Source Code

*Tabel 2. Modul 1 Source Code Soal 2*

```
1.  #include <iostream>
2.  #include <conio.h>
3.  #include <string>
4.  using namespace std;
5.
6.  const int max_stack = 10;
7.  string PILIHAN, data;
8.  int PIL;
9.
10. struct Stack {
11.     int atas;
12.     string data[max_stack];
13. } Tumpuk;
14.
15. int penuh() {
16.     if (Tumpuk.atas == max_stack - 1)
17.         return 1;
18.     else
19.         return 0;
20. }
21.
22. int kosong() {
23.     if (Tumpuk.atas == -1)
24.         return 1;
25.     else
26.         return 0;
27. }
28.
29. void input()
30. {
31.     if (penuh() == 0){
32.         cout << "Masukkan data: ";
33.         cin >> data;
```

```

34.         Tumpuk.atas++;
35.         Tumpuk.data[Tumpuk.atas] = data;
36.         cout << "Data " << Tumpuk.data[Tumpuk.atas] << " Masuk
Ke Stack" << endl;
37.     }else if (penuh() == 0) {
38.         cout << "Masukkan data: ";
39.         cin >> data;
40.         Tumpuk.atas++;
41.         Tumpuk.data[Tumpuk.atas] = data;
42.         cout << "Data " << Tumpuk.data[Tumpuk.atas] << " masuk
ke stack\n";
43.     }
44.     else
45.         cout << "Tumpukan Penuh" << endl;
46. }
47.
48.
49. void hapus() {
50.     if (kosong() == 0) {
51.         cout << "Data \"" << Tumpuk.data[Tumpuk.atas] << "\"
teratas sudah terambil\n";
52.         Tumpuk.atas--;
53.     }
54.     else {
55.         cout << "Data kosong\n";
56.     }
57. }
58.
59. void tampil() {
60.     if (kosong() == 0) {
61.         for (int i = Tumpuk.atas; i >= 0; i--) {
62.             cout << "Tumpukan ke-" << i << " = \"" <<
Tumpuk.data[i] << "\"\n" << endl;
63.         }
64.     }
65.     else {

```

```

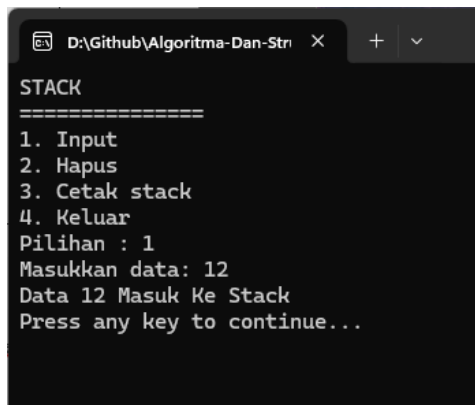
66.         cout << "Tumpukan kosong\n";
67.     }
68. }
69.
70. void bersih() {
71.     Tumpuk.atas = -1;
72.     cout << "Tumpukan kosong!\n";
73. }
74.
75. int main() {
76.     Tumpuk.atas = -1;
77.     do {
78.         cout << "STACK" << endl;
79.         cout << "=====" << endl;
80.         cout << "1. Input" << endl;
81.         cout << "2. Hapus" << endl;
82.         cout << "3. Cetak stack" << endl;
83.         cout << "4. Keluar" << endl;
84.         cout << "Pilihan : ";
85.         cin >> PILIHAN;
86.         PIL = stoi(PILIHAN);
87.
88.         switch (PIL) {
89.             case 1:
90.                 input();
91.                 break;
92.             case 2:
93.                 hapus();
94.                 break;
95.             case 3:
96.                 tampil();
97.                 break;
98.             case 4:
99.                 cout << "TERIMA KASIH" << endl;
100.                break;
101.            default:

```



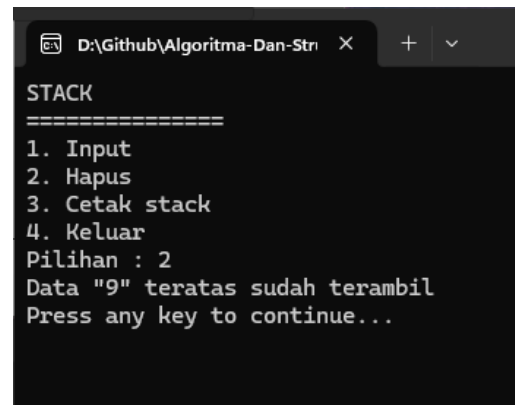
```
102.         cout << "Pilihan tidak valid" << endl;
103.         break;
104.     }
105.     cout << "Press any key to continue..." << endl;
106.     getch();
107.     system("cls");
108. } while (PIL != 4);
109. return 0;
110. }
111. #include <iostream>
112. #include <conio.h>
```

## B. Output Program



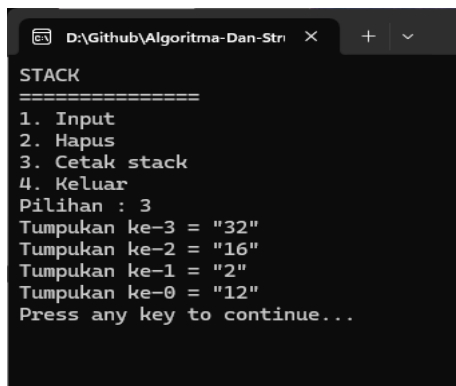
```
STACK
=====
1. Input
2. Hapus
3. Cetak stack
4. Keluar
Pilihan : 1
Masukkan data: 12
Data 12 Masuk Ke Stack
Press any key to continue...
```

Gambar 4. Modul 2 Output Opsi Input Saol 2



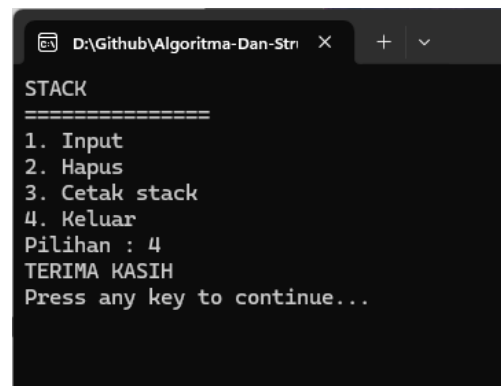
```
STACK
=====
1. Input
2. Hapus
3. Cetak stack
4. Keluar
Pilihan : 2
Data "9" teratas sudah diambil
Press any key to continue...
```

Gambar 5. Modul 2 Output Opsi Hapus Saol 2



```
STACK
=====
1. Input
2. Hapus
3. Cetak stack
4. Keluar
Pilihan : 3
Tumpukan ke-3 = "32"
Tumpukan ke-2 = "16"
Tumpukan ke-1 = "2"
Tumpukan ke-0 = "12"
Press any key to continue...
```

Gambar 6. Modul 2 Output Opsi Cetak Saol 2



```
STACK
=====
1. Input
2. Hapus
3. Cetak stack
4. Keluar
Pilihan : 4
TERIMA KASIH
Press any key to continue...
```

Gambar 7. Modul 2 Output Opsi Keluar Saol 2

## C. Pembahasan

### Penambahan Library dan Namespace

Bagian ini mendeklarasikan library yang dibutuhkan oleh program:

- `<iostream>`: Diperlukan untuk operasi input/output standar seperti `cin` dan `cout`
- `<conio.h>`: Library yang menyediakan fungsi `getch()` untuk menahan layar sampai pengguna menekan tombol
- `<string>`: Library yang menambahkan dukungan untuk manipulasi string dalam C++.

Penggunaan using namespace std; memungkinkan program menggunakan fungsi-fungsi dari namespace standard (std) tanpa harus menuliskan prefiks "std::" setiap kali, seperti std::cout atau std::string.

### **Deklarasi Variabel global**

Bagian ini mendefinisikan konstanta dan variabel global yang digunakan dalam program:

- a. `const int max_stack = 10;` Mendefinisikan ukuran maksimum stack sebagai konstanta bernilai 10.
- b. `string PILIHAN;` Variabel untuk menyimpan input pilihan menu dari pengguna dalam bentuk string.
- c. `int PIL;` Variabel untuk menyimpan pilihan menu yang sudah dikonversi ke integer untuk digunakan dalam struktur kontrol program.
- d. `string data;` Variabel untuk menyimpan data yang akan dimasukkan ke dalam stack.

### **Struktur Data Stack**

Bagian ini mendefinisikan struktur data stack dan variabel instance-nya:

- a. `struct Stack`: Mendefinisikan tipe data struktur bernama Stack yang terdiri dari:
  - 1) `int atas`: Variabel yang berfungsi sebagai pointer/indeks untuk menunjukkan posisi elemen teratas pada stack.
  - 2) `string data[max_stack]`: Array dengan ukuran `max_stack` untuk menyimpan elemen-elemen stack dalam bentuk string.
- b. `Tumpuk`: Variabel instance dari struktur Stack yang akan digunakan untuk operasi stack dalam program.

### **Fungsi penuh()**

Fungsi `penuh()` bertugas untuk memeriksa apakah stack dalam keadaan penuh. Fungsi ini mengembalikan nilai:

- a. 1 (true) jika stack penuh.
- b. 0 (false) jika stack tidak penuh.

Logika yang digunakan adalah dengan memeriksa apakah indeks Tumpuk.atas sudah mencapai nilai maksimum stack dikurangi 1 (max - 1).

### **Fungsi kosong()**

Fungsi kosong() bertugas untuk memeriksa apakah stack dalam keadaan kosong. Fungsi ini mengembalikan nilai:

- a. 1 (true) jika stack kosong.
- b. 0 (false) jika stack tidak kosong.

Logika yang digunakan adalah dengan memeriksa apakah indeks Tumpuk.atas sudah mencapai nilai maksimum stack dikurangi 1 (max - 1).

### **Fungsi input()**

Fungsi input() digunakan untuk memasukkan data ke dalam stack. Fungsi ini memiliki beberapa kondisi:

- a. Jika stack tidak penuh (penuh() == 0), maka data di input lalu nilai atas ditambah 1 dan data dimasukkan.
- b. Jika stack tidak penuh (penuh() == 0), maka data juga di input lalu nilai atas ditambah 1 dan data dimasukkan.
- c. Jika kondisi lainnya (yaitu stack penuh), maka ditampilkan pesan "Tumpukan Penuh".

Logika yang digunakan adalah dengan memeriksa apakah nilai dari fungsi penuh bernilai false.

### **Fungsi hapus()**

Fungsi hapus() bertugas untuk menghapus elemen teratas dari stack. Fungsi ini:

- a. Jika tidak kosong, maka indeks pointer Tumpuk.atas dikurangi 1, yang secara efektif menghapus elemen teratas.
- b. Jika kosong, maka ditampilkan pesan "Data Kosong".

Fungsi ini tidak benar-benar menghapus data secara fisik dari array, melainkan hanya mengubah indeks teratas yang dapat diakses.

### **Fungsi tampil ()**

Fungsi tampil() bertugas untuk menampilkan seluruh isi stack dari atas ke bawah. Fungsi ini:

- a. Jika tidak kosong, menampilkan seluruh elemen stack mulai dari indeks teratas hingga indeks terbawah (0) .
- b. Jika kosong, menampilkan pesan "Tumpukan Kosong".

Implementasi ini sesuai dengan prinsip LIFO (Last In First Out) dari stack, di mana elemen teratas ditampilkan terlebih dahulu.

### **Fungsi bersih()**

Fungsi bersih() bertugas untuk mengosongkan stack. Fungsi ini:

- a. Menetapkan nilai pointer Tumpuk.atas menjadi -1, yang mengindikasikan stack kosong
- b. Menampilkan pesan konfirmasi "Tumpukan Kosong!"

Sama seperti fungsi hapus, fungsi ini tidak benar-benar menghapus data secara fisik dari array, melainkan hanya mengatur ulang nilai atas menjadi -1 sehingga stack dianggap kosong dan siap diisi kembali.

### SOAL 3

Cobalah contoh program berikut, running dan analisis hasilnya!

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
#define n 10
using namespace std;

void INSERT();
void DELETE();
void CETAKLAYAR();
void Inisialisasi();
void RESET();
int PIL,F,R;
char PILIHAN [1],HURUF;
char Q[n];
int main ( )
{
    Inisialisasi();
    do
    {
        cout<<"QUEUE"<<endl;
        cout<<"===== "<<endl;
        cout<<"1. INSERT"<<endl;
        cout<<"2. DELETE"<<endl;
        cout<<"3. CETAK QUEUE"<<endl;
        cout<<"4. QUIT"<<endl;
        cout<<"PILIHAN : "; cin>>PILIHAN;
        PIL=atoi(PILIHAN);

        switch (PIL)
        {
            case 1:
                INSERT ();
                break;
            case 2:
                DELETE();
                break;
            case 3:
                CETAKLAYAR ();
                break;
            default:
                cout<<"TERIMA KASIH"<<endl;
                break;
        }
        cout<<"press any key to continue"<<endl;
        getch();
        system("cls");
    }
    while (PIL<4);
}
```

Gambar 8. Modul 2 Soal 3

## A. Source Code

Tabel 3. Modul 1 Source Code Soal 3

1.	#include <iostream>
2.	#include <conio.h>
3.	#include <stdlib.h>
4.	#define n 10
5.	using namespace std;
6.	
7.	int PIL, F, R;
8.	char PILIHAN[1], HURUF;
9.	char Q[n];
10.	
11.	void Inisialisasi() {
12.	F = 0;
13.	R = -1;
14.	}
15.	
16.	bool isFull() {
17.	return R == n - 1;
18.	}
19.	
20.	bool isEmpty() {
21.	return F > R;
22.	}
23.	
24.	void INSERT() {
25.	if (isFull()) {
26.	cout << "Queue Penuh!" << endl;
27.	} else {
28.	cout << "Masukkan 1 huruf: ";
29.	cin >> HURUF;
30.	R++;
31.	Q[R] = HURUF;
32.	cout << "Data '" << HURUF << "' berhasil dimasukkan ke dalam queue!" << endl;

```

33.     }
34. }
35.
36. void DELETE() {
37.     if (isEmpty()) {
38.         cout << "Queue Kosong!" << endl;
39.     } else {
40.         cout << "Data '" << Q[F] << "' telah dihapus dari queue"
<< endl;
41.         F++;
42.         if (isEmpty()) {
43.             Inisialisasi();
44.             cout << "Queue telah direset" << endl;
45.         }
46.     }
47. }
48.
49. void CETAKLAYAR() {
50.     if (isEmpty()) {
51.         cout << "Queue Kosong!" << endl;
52.     } else {
53.         cout << "Isi Queue: ";
54.         for (int i = F; i <= R; i++) {
55.             cout << Q[i] << " ";
56.         }
57.         cout << endl;
58.         cout << "Front: " << F << ", Rear: " << F << endl;
59.     }
60. }
61.
62. int main() {
63.     Inisialisasi();
64.     do {
65.         cout << "QUEUE" << endl;
66.         cout << "======" << endl;
67.         cout << "1. INSERT" << endl;

```

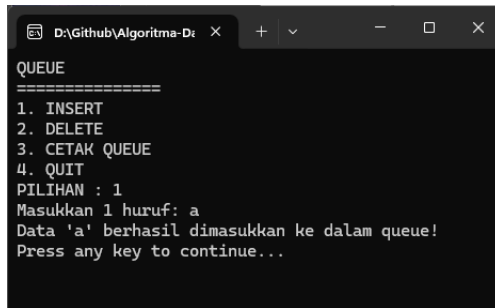


```

68.         cout << "2. DELETE" << endl;
69.         cout << "3. CETAK QUEUE" << endl;
70.         cout << "4. QUIT" << endl;
71.         cout << "PILIHAN : ";
72.         cin >> PILIHAN;
73.         PIL = atoi(PILIHAN);
74.
75.         switch (PIL) {
76.             case 1:
77.                 INSERT();
78.                 break;
79.             case 2:
80.                 DELETE();
81.                 break;
82.             case 3:
83.                 CETAKLAYAR();
84.                 break;
85.             default:
86.                 cout << "TERIMA KASIH" << endl;
87.                 break;
88.         }
89.
90.         cout << "Press any key to continue..." << endl;
91.         getch();
92.         system("cls");
93.     } while (PIL < 4);
94.     return 0;

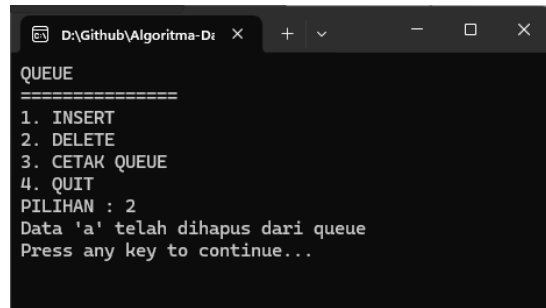
```

## B. Output Program



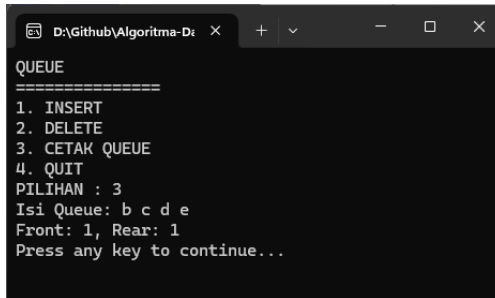
```
QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. QUIT
PILIHAN : 1
Masukkan 1 huruf: a
Data 'a' berhasil dimasukkan ke dalam queue!
Press any key to continue...
```

Gambar 9. Modul 2 Output Opsi Insert Saol 3



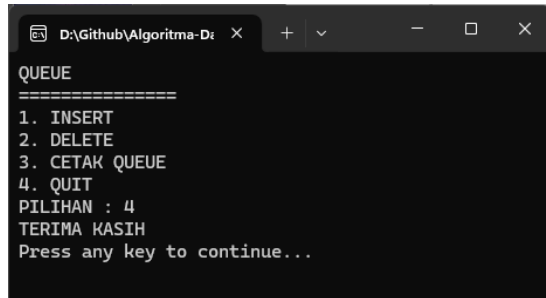
```
QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. QUIT
PILIHAN : 2
Data 'a' telah dihapus dari queue
Press any key to continue...
```

Gambar 10. Modul 2 Output Opsi Delete Saol 3



```
QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. QUIT
PILIHAN : 3
Isi Queue: b c d e
Front: 1, Rear: 1
Press any key to continue...
```

Gambar 11. Modul 2 Output Opsi Cetak Saol 3



```
QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. QUIT
PILIHAN : 4
TERIMA KASIH
Press any key to continue...
```

Gambar 12. Modul 2 Output Opsi Quit Saol 3

## C. Pembahasan

### Penambahan Library dan Namespace

Bagian ini mendeklarasikan library yang dibutuhkan oleh program:

- `<iostream>`: Untuk operasi input/output standar seperti `cin` dan `cout`
- `<conio.h>`: Menyediakan fungsi `getch()` untuk menahan tampilan layar
- `<stdlib.h>`: Menyediakan fungsi `system()` untuk membersihkan layar
- `#define n 10`: Mendefinisikan konstanta `n` dengan nilai 10 yang digunakan sebagai ukuran maksimum queue
- `using namespace std`: Memungkinkan penggunaan fungsi dari namespace standard tanpa prefiks `std::`

### **Deklarasi Variabel global**

Bagian ini mendefinisikan konstanta dan variabel global yang digunakan dalam program:

- a. PIL: Variabel integer untuk menyimpan pilihan menu setelah konversi
- b. F (Front): Indeks untuk menunjukkan elemen terdepan dalam queue
- c. R (Rear): Indeks untuk menunjukkan elemen terakhir dalam queue
- d. PILIHAN[1]: Array karakter untuk menyimpan input pilihan menu (terbatas hanya 1 karakter)
- e. HURUF: Variabel karakter untuk menyimpan data yang akan dimasukkan ke queue
- f. Q[n]: Array dengan ukuran n untuk menyimpan elemen-elemen queue.

### **Fungsi Inisialisasi**

Fungsi ini menginisialisasi queue dengan menetapkan nilai awal untuk indeks front (F) dan rear (R):

- a. Front (F) diatur ke 0, menunjukkan posisi awal untuk pengambilan data
- b. Rear (R) diatur ke -1, menandakan queue kosong (belum ada elemen yang dimasukkan).

inisialisasi standar untuk queue linear, di mana indeks rear awalnya diatur lebih rendah dari front untuk menandakan kekosongan.

### **isFull() dan isEmpty()**

Dua fungsi boolean untuk memeriksa status queue:

- a. isFull(): Mengembalikan true jika rear (R) telah mencapai indeks terakhir array (n-1)
- b. isEmpty(): Mengembalikan true jika front (F) lebih besar dari rear (R).

### **Fungsi INSERT()**

Fungsi INSERT() digunakan untuk memasukkan data ke dalam queue. Fungsi ini memiliki beberapa kondisi:

- a. Jika stack penuh (isFull() == 1), maka ditampilkan pesan "Queue Penuh!".

- b. Jika kondisi lainnya (yaitu stack tidak penuh), "maka HURUF di input lalu nilai rear(R) ditambah 1 dan HURUF dimasukkan ke posisi rear(R), pesan konfirmasi akan ditampilkan dengan menunjukkan data yang dimasukkan.

### **Fungsi DELETE ()**

Fungsi DELETE() bertugas untuk menghapus elemen teratas dari queue.

Fungsi ini:

- a. Jika tidak kosong, maka indeks pointer Tumpuk.atas dikurangi 1, yang secara efektif menghapus elemen teratas.
- b. Jika kosong, maka ditampilkan pesan "Data Kosong".

Fungsi ini tidak benar-benar menghapus data secara fisik dari array, melainkan hanya mengubah indeks teratas yang dapat diakses.

### **Fungsi CETAKLAYAR()**

Fungsi CETAKLAYAR() bertugas untuk menampilkan seluruh isi queue.

Fungsi ini:

- a. Jika kosong, menampilkan pesan "Tumpukan Kosong".
- b. Jika tidak kosong, menampilkan seluruh elemen queue mulai dari front hingga rear.

Penambahan informasi tentang nilai indeks front dan rear juga ditambahkan, sangat berguna untuk memahami status internal queue.

### **Fungsi main()**

Fungsi utama yang mengontrol alur program:

- a. Memanggil Inisialisasi() untuk menyiapkan queue
- b. Menampilkan menu interaktif dengan 4 pilihan
- c. Membaca input pilihan pengguna dan mengkonversinya ke integer dengan atoi()
- d. Menjalankan fungsi yang sesuai berdasarkan pilihan menggunakan struktur switch
- e. Menunggu pengguna menekan tombol untuk melanjutkan

- f. Membersihkan layar sebelum menampilkan menu kembali
- g. Melanjutkan loop selama pilihan pengguna kurang dari 4

## MODUL 3 : SINGLE LINK LIST

### SOAL 1

Cobalah program berikut, running, simpan program, dan screenshoot hasil running !

```
1  #include <conio.h>
2  #include <iostream>
3  #include <stdlib.h>
4
5  using namespace std;
6
7  typedef struct TNode {
8      string data;
9      TNode *next;
10 };
11
12 TNode *head, *tail;
13
14 int pill;
15 char pilihan[2];
16 string databaru, data;
17
18 void init();
19 int isEmpty();
20
21 void tambahDepan();
22 void tambahBelakang();
23 void hapusDepan();
24 void hapusBelakang();
25 void tampilkan();
26 void reset();
27 void cariData();
28 void hapusData();
29 void sisipkanSebelum();
30 void sisipkanSetelah();
31
32 int main()
33 {
34     do {
35         cout<<"Single Linked List Circular (SLIC)"<<endl;
36         cout<<"-----"<<endl;
37         cout<<"1. Tambah Depan"<<endl;
38         cout<<"2. Tambah Belakang"<<endl;
39         cout<<"3. Hapus Depan"<<endl;
40         cout<<"4. Hapus Belakang"<<endl;
41         cout<<"5. Tampilkan Data"<<endl;
42         cout<<"6. Hapus Semua Elemen"<<endl;
43         cout<<"7. Cari Data"<<endl;
44         cout<<"8. Hapus Setiap Data Tertentu"<<endl;
45         cout<<"9. Sisipkan Node/Data Baru Sebelum Data Tertentu"<<endl;
46         cout<<"10. Sisipkan Node/Data Baru Setelah Data Tertentu"<<endl;
47         cout<<"11. Quit"<<endl;
48         cout<<"Pilihan : ?"<<endl;
49         cin>>pilihan;
50         pill=atoi(pilihan);
51
52         switch(pill) {
53             case 1:
54                 tambahDepan();
55                 break;
56             case 2:
57                 tambahBelakang();
58                 break;
59             case 3:
60                 hapusDepan();
61                 cout<<"Data \""<<data<<"\" yang berada di depan telah berhasil dihapus."<<endl;
62                 break;
63             case 4:
64                 hapusBelakang();
65                 cout<<"Data \""<<data<<"\" yang berada di belakang telah berhasil dihapus."<<endl;
66                 break;
67             case 5:
68                 tampilkan();
69                 break;
70             case 6:
71                 reset();
72                 break;
73             case 7:
74                 cariData();
75                 break;
76             case 8:
77                 hapusData();
78                 break;
79             case 9:
80                 sisipkanSebelum();
81                 break;
82             case 10:
83                 sisipkanSetelah();
84                 break;
85             default:
86                 cout<<"\nTERIMA KASIH"<<endl;
87                 cout<<"Program was made by Hana (HDM)."<<endl;
88         }
89
90         cout<<"\nPress any key to continue!"<<endl;
91         getch();
92         system("cls");
93     } while (pill<11);
94 }
```

```

95 }
96
97 void init(){
98     head = NULL;
99     tail = NULL;
100 }
101
102 int isEmpty() {
103     if(head == NULL) return 1;
104     else return 0;
105 }
106
107 void tambahDepan() {
108     cout<<"Masukkan data : ";
109     TNode *baru;
110     baru = new TNode;
111     cin>>dataBaru;
112     baru->data = dataBaru;
113     baru->next = baru;
114
115     if(isEmpty() == 1) {
116         head = baru;
117         tail = baru;
118     } else {
119         baru->next = head;
120         head = baru;
121         tail->next = head;
122     }
123     cout << "Data \'"<<dataBaru<<"\" berhasil dimasukkan di bagian depan."<<endl;
124 }
125
126 void tambahBelakang() {
127     cout<<"Masukkan data : ";
128     TNode *baru;
129     baru = new TNode;
130     cin>>dataBaru;
131     baru->data = dataBaru;
132     baru->next = baru;
133
134     if(isEmpty() == 1) {
135         head = baru;
136         tail = baru;
137     } else {
138         tail->next = baru;
139         tail = baru;
140         tail->next = head;
141     }
142     cout << "Data \'"<<dataBaru<<"\" berhasil dimasukkan di bagian belakang."<<endl;
143 }
144
145 void hapusDepan() {
146     if(isEmpty() == 0) {
147         TNode *hapus;
148         hapus = head;
149         data = hapus->data;
150
151         if(head != tail) {
152             head = head->next;
153             tail->next = head;
154         } else {
155             init();
156         }
157
158         delete hapus;
159     } else cout<<"Tidak terdapat data pada linked list."<<endl;
160 }
161
162 void hapusBelakang() {
163     if(isEmpty() == 0) {
164         TNode *hapus, *newTail;
165         hapus = tail;
166         data = hapus->data;
167
168         if(head != tail) {
169             newTail = head;
170             while(newTail->next != tail) {
171                 newTail = newTail->next;
172             }
173             tail = newTail;
174             tail->next = head;
175         } else {
176             init();
177         }
178
179         delete hapus;
180     } else cout<<"Tidak terdapat data pada linked list."<<endl;
181 }
182
183 void tampilkan() {
184     if(isEmpty() == 0) {
185         TNode *bantu;

```

```

186     bantu = head;
187
188     do {
189         cout<<"bantu->data<<" ";
190         bantu = bantu->next;
191     } while(bantu != head);
192     cout<<endl;
193 } else cout<<"Tidak terdapat data pada Linked List."<<endl;
194 }
195
196 void reset() {
197     if(isEmpty() == 0) {
198         TNode *bantu, *hapus;
199         bantu = head;
200
201         do {
202             hapus = bantu;
203             bantu = bantu->next;
204             delete hapus;
205         } while(bantu != head);
206
207         init();
208         cout<<"seluruh elemen pada Linked List telah dibersihkan."<<endl;
209     } else cout<<"Tidak terdapat data pada Linked List."<<endl;
210 }
211
212 void cariData() {
213     if(isEmpty() == 0) {
214         string cari;
215         cout<<"Masukkan data yang ingin dicari : ";
216         cin>>cari;
217
218         TNode *bantu, *hapus, *newTail, *bantuJampilkan;
219         bool apaDitemukan = false;
220
221         bantu = head;
222
223         do {
224             if(cari == bantu->data){
225                 cout<<"Selanjut data yang berada di dalam tanda kurung siku ([...]) "
226                     <<"merupakan data yang anda cari."<<endl;
227                 cout<<"Linked List : ";
228                 bantuJampilkan = head;
229
230                 do {
231                     if(cari == bantuJampilkan->data)
232                         cout<<"["<<bantuJampilkan->data<<"] ";
233                     else
234                         cout<<bantuJampilkan->data<<" ";
235                     bantuJampilkan = bantuJampilkan->next;
236                 } while(bantuJampilkan != head);
237
238                 apaDitemukan = true;
239                 cout<<endl;
240                 break;
241             }
242             bantu = bantu->next;
243         } while(bantu != head);
244
245         if(apaDitemukan == false)
246             cout<<"Data \""<<cari<<"\" tidak ditemukan pada Linked List."<<endl;
247     } else cout<<"Tidak terdapat data pada Linked List."<<endl;
248 }
249
250 void hapusData() {
251     if(isEmpty() == 0) {
252         string cari;
253         cout<<"Masukkan data yang ingin dihapus : ";
254         cin>>cari;
255
256         TNode *bantu, *sebelum, *hapus[255], *bantu2;
257         int hitung = 0;
258         bool apaDitemukan = false;
259
260         bantu = head;
261
262         do {
263             bantu2 = bantu;
264             if(cari == bantu->data){
265                 hapus[hitung++] = bantu;
266                 apaDitemukan = true;
267                 if(bantu != head && bantu != tail) {
268                     sebelum->next = bantu->next;
269                     bantu2 = sebelum;
270                 }
271             }
272             sebelum = bantu2;
273             bantu = bantu->next;
274         } while(bantu != head);
275
276         if(apaDitemukan == true) {
277             for(int i = 0; i < hitung; i++) {

```



```

278         if(hapus[i] == head){
279             hapusDepan();
280         } else if(hapus[i] == tail) {
281             hapusBelakang();
282         } else {
283             delete hapus[i];
284         }
285     }
286
287     cout<<"Getap data \\"<<endl; yang terdapat pada linked list telah dihapus";
288 } else cout<<"Data \\"<<endl; tidak ditemukan pada linked list."<<endl;
289
290 } else cout<<"Tidak terdapat data pada linked list."<<endl;
291
292
293 void sisipkanSebelum() {
294     if(isEmpty() == 0) {
295         TNode *bantu, *sebelum;
296         string nextData;
297         bool apaAda;
298
299         bantu = head;
300         sebelum = tail;
301
302         cout<<"Sisipkan data baru sebelum data : ";
303         cin>>nextData;
304
305         do {
306             if(nextData == bantu->data){
307                 apaAda = true;
308                 break;
309             } else {
310                 sebelum = bantu;
311                 bantu = bantu->next;
312             }
313         } while(bantu != head);
314
315         if(apaAda==true) {
316             cout<<"Masukkan data yang ingin ditambahkan : ";
317             cin>>dataBaru;
318
319             TNode *baru;
320             baru = new TNode;
321
322             baru->data = dataBaru;
323             baru->next = bantu;
324
325             sebelum->next = baru;
326
327             if(bantu == head){
328                 head = baru;
329             }
330
331             cout << "Data \\"<<dataBaru<<" berhasil disisipkan sebelum data \\"<<nextData<<"."<<endl;
332         } else {
333             cout<<"Tidak terdapat data \\"<<nextData<<" pada linked list."<<endl;
334         }
335     } else cout<<"Tidak terdapat data pada linked list."<<endl;
336 }
337
338 void sisipkanSetelah() {
339     if(isEmpty() == 0) {
340         TNode *bantu;
341         string prevData;
342         bool apaAda;
343
344         bantu = head;
345
346         cout<<"Sisipkan data baru setelah data : ";
347         cin>>prevData;
348
349         do {
350             if(prevData == bantu->data){
351                 apaAda = true;
352                 break;
353             } else {
354                 bantu = bantu->next;
355             }
356         } while(bantu != head);
357
358         if(apaAda==true) {
359             cout<<"Masukkan data yang ingin ditambahkan : ";
360             cin>>dataBaru;
361
362             TNode *baru;
363             baru = new TNode;
364
365             baru->data = dataBaru;
366             baru->next = bantu->next;
367
368             bantu->next = baru;
369
370             if(bantu == tail){
371                 tail = baru;
372             }
373
374             cout << "Data \\"<<dataBaru<<" berhasil disisipkan setelah data \\"<<prevData<<"."<<endl;
375         } else {
376             cout<<"Tidak terdapat data \\"<<prevData<<" pada linked list."<<endl;
377         }
378     } else cout<<"Tidak terdapat data pada linked list."<<endl;
379 }
380

```

Gambar 13. Modul 3 Soal 1

## A. Source Code

*Tabel 4. Modul 3 Source Code Soal 1*

1.	#include <conio.h>
2.	#include <iostream>
3.	#include <stdlib.h>
4.	
5.	using namespace std;
6.	
7.	typedef struct TNode {
8.	string data;
9.	TNode *next;
10.	};
11.	
12.	TNode *head, *tail;
13.	
14.	int pil;
15.	char pilihan[2];
16.	string dataBaru, data;
17.	
18.	void init();
19.	int isEmpty();
20.	
21.	void tambahDepan();
22.	void tambahBelakang();
23.	void hapusDepan();
24.	void hapusBelakang();
25.	void tampilkan();
26.	void reset();
27.	void cariData();
28.	void hapusData();
29.	void sisipkanSebelum();
30.	void sisipkanSetelah();
31.	
32.	int main()
33.	{

```

34.     do {
35.         cout<<"Single Linked List Circular (SLLC)"<<endl;
36.         cout<<"====="<<endl;
37.         cout<<"1. Tambah Depan"<<endl;
38.         cout<<"2. Tambah Belakang"<<endl;
39.         cout<<"3. Hapus Depan"<<endl;
40.         cout<<"4. Hapus Belakang"<<endl;
41.         cout<<"5. Tampilkan Data"<<endl;
42.         cout<<"6. Hapus Semua Elemen"<<endl;
43.         cout<<"7. Cari Data"<<endl;
44.         cout<<"8. Hapus Setiap Data Tertentu"<<endl;
45.         cout<<"9.   Sisipkan   Node/Data   Baru   Sebelum   Data
Tertentu"<<endl;
46.         cout<<"10.  Sisipkan   Node/Data   Baru   Setelah   Data
Tertentu"<<endl;
47.         cout<<"11. Quit"<<endl;
48.         cout<<"Pilihan :"<<endl;
49.         cin>>pilihan;
50.         pil=atoi(pilihan);
51.
52.         switch(pil) {
53.             case 1:
54.                 tambahDepan();
55.                 break;
56.             case 2:
57.                 tambahBelakang();
58.                 break;
59.             case 3:
60.                 hapusDepan();
61.                 cout<<"data \"<<data<<"\" yang berada di depan telah
berhasil dihapus."<<endl;
62.                 break;
63.             case 4:
64.                 hapusBelakang();
65.                 cout<<"data \"<<data<<"\" yang berada di belakang
telah berhasil dihapus."<<endl;

```

```

66.         break;
67.     case 5:
68.         tampilkan();
69.         break;
70.     case 6:
71.         reset();
72.         break;
73.     case 7:
74.         cariData();
75.         break;
76.     case 8:
77.         hapusData();
78.         break;
79.     case 9:
80.         sisipkanSebelum();
81.         break;
82.     case 10:
83.         sisipkanSetelah();
84.         break;
85.     default:
86.         cout<<"\nTERIMA KASIH"<<endl;
87.         cout<<"Program was made by Indra Suryadilga
(2410817310014)."<<endl;
88.     }
89.
90.     cout<<"\nPress any key to continiue!"<<endl;
91.     getch();
92.     system("cls");
93.
94.     } while (pil<11);
95. }
96.
97. void init() {
98.     head = NULL;
99.     tail = NULL;
100. }

```

```

101.
102. int isEmpty() {
103.     if(head==NULL) return 1;
104.     else return 0;
105. }
106.
107. void tambahDepan() {
108.     cout<<"Masukkan data : ";
109.     TNode *baru;
110.     baru = new TNode;
111.     cin>>dataBaru;
112.     baru->data = dataBaru;
113.     baru->next = baru;
114.
115.     if(isEmpty()==1) {
116.         head = baru;
117.         tail = baru;
118.     } else {
119.         baru->next = head;
120.         head = baru;
121.         tail->next = head;
122.     }
123.     cout<<"Data \""<<dataBaru<<"\" berhasil dimasukan di bagian
depan."<<endl;
124. }
125.
126. void tambahBelakang() {
127.     cout<<"Masukkan data : ";
128.     TNode *baru;
129.     baru = new TNode;
130.     cin>>dataBaru;
131.     baru->data = dataBaru;
132.     baru->next = baru;
133.
134.     if(isEmpty()==1) {
135.         head = baru;

```

```

136.     tail = baru;
137. } else {
138.     tail->next = baru;
139.     tail = baru;
140.     tail->next = head;
141. }
142.     cout<<"Data \""<<dataBaru<<"\" berhasil dimasukan di bagian
    belaknag."<<endl;
143. }
144.
145. void hapusDepan() {
146.     if(isEmpty()==0) {
147.         TNode *hapus;
148.         hapus = head;
149.         data = hapus->data;
150.
151.         if(head != tail) {
152.             head = head->next;
153.             tail->next = head;
154.         } else {
155.             init();
156.         }
157.
158.         delete hapus;
159.     } else cout<<"Tidak terdapat data pada Linked List."<<endl;
160. }
161.
162. void hapusBelakang() {
163.     if(isEmpty()==0) {
164.         TNode * hapus, *newTail;
165.         hapus = tail;
166.         data = hapus->data;
167.
168.         if(head != tail) {
169.             newTail = head;
170.             while(newTail->next != tail) {

```

```

171.         newTail = newTail->next;
172.     }
173.     tail = newTail;
174.     tail->next = head;
175. } else {
176.     init();
177. }
178.
179.     delete hapus;
180. } else cout<<"Tidak terdapat data pada Linked List."<<endl;
181. }
182.
183. void tampilkan() {
184.     if(isEmpty()==0) {
185.         TNode * bantu;
186.         bantu = head;
187.
188.         do {
189.             cout<<bantu->data<<' ';
190.             bantu = bantu->next;
191.         } while(bantu != head);
192.         cout<<endl;
193.     } else cout<<"Tidak terdapat data pada Linked List."<<endl;
194. }
195.
196. void reset() {
197.     if(isEmpty() == 0) {
198.         TNode *bantu, *hapus;
199.         bantu = head;
200.
201.         do {
202.             hapus = bantu;
203.             bantu = bantu->next;
204.             delete hapus;
205.         } while(bantu != head);
206.

```

```

207.     init();
208.     cout<<"Seluruh elemen pada Linked List telah
dibersihkan."<<endl;
209.     }else cout<<"Tidak terdapat data pada Linked List."<<endl;
210. }
211.
212. void cariData() {
213.     if(isEmpty() == 0) {
214.         string cari;
215.         cout<<"Masukan data yang ingin dicari : ";
216.         cin>>cari;
217.
218.         TNode *bantu, *hapus, *newTail, *bantuTampilkan;
219.         bool apaDitemukan = false;
220.
221.         bantu = head;
222.
223.         do {
224.             if(cari == bantu->data){
225.                 cout<<"Setiap data yang berada di dalam tanda kurung
siku([..]) "
226.                     <<"merupakan data yang dicari."<<endl;
227.                 cout<<"Linked List : ";
228.                 bantuTampilkan = head;
229.
230.                 do {
231.                     if(cari == bantuTampilkan->data)
232.                         cout<<"["<<bantuTampilkan->data<<"] ";
233.                     else
234.                         cout<<bantuTampilkan->data<<' ';
235.                     bantuTampilkan = bantuTampilkan->next;
236.                 } while(bantuTampilkan != head);
237.
238.                 apaDitemukan = true;
239.                 cout<<endl;
240.                 break;

```



```

241.     }
242.     bantu = bantu->next;
243. } while(bantu != head);
244.
245. if(apaDitemukan ==false)
246.     cout<<"Data \""<<cari<<"\" tidak ditemukan pada Linked
List."<<endl;
247. } else cout<<"Tidak terdapat data pada Linked List."<<endl;
248. }
249.
250. void hapusData() {
251.     if(isEmpty() == 0) {
252.         string cari;
253.         cout<<"Masukkan data yang ingin dihapus : ";
254.         cin>>cari;
255.
256.         TNode *bantu, *sebelum, *hapus[255], *bantu2;
257.         int hitung = 0;
258.         bool apaDitemukan = false;
259.
260.         bantu = head;
261.
262.         do {
263.             bantu2 = bantu;
264.             if(cari == bantu->data){
265.                 hapus[hitung++] = bantu;
266.                 apaDitemukan = true;
267.                 if(bantu != head && bantu != tail) {
268.                     sebelum->next = bantu->next;
269.                     bantu2 = sebelum;
270.                 }
271.             }
272.             bantu = bantu->next;
273.             } while(bantu != head);
274.
275.

```

```

276.     if(apaDitemukan == true) {
277.         for(int i = 0; i < hitung; i++) {
278.             if(hapus[i] == head) {
279.                 hapusDepan();
280.             } else if(hapus[i] == tail) {
281.                 hapusBelakang();
282.             } else {
283.                 delete hapus[i];
284.             }
285.         }
286.
287.         cout<<"Setipa data \""<<cari<<"\" yang terdapat pada
Linked List telah dihapus";
288.     } else cout<<"Data \""<<cari<<"\" tidak ditemukan pada
Linked List."<<endl;
289.
290.     } else cout<<"Tidak terdapat data pada Linked List."<<endl;
291. }
292.
293. void sisipkanSebelum() {
294.     if(isEmpty() == 0) {
295.         TNode *bantu, *sebelum;
296.         string nextData;
297.         bool apaAda;
298.
299.         bantu = head;
300.         sebelum = tail;
301.
302.         cout<<"Sisipkan data baru sebelum data : ";
303.         cin>>nextData;
304.
305.         do {
306.             if(nextData == bantu->data) {
307.                 apaAda = true;
308.                 break;
309.             } else {

```

```

310     sebelum = bantu;
311     bantu = bantu->next;
312 }
313 } while(bantu != head);
314
315 if(apaAda == true) {
316     cout<<"Masukan data yang diinginkan : ";
317     cin>>dataBaru;
318
319     TNode *baru;
320     baru = new TNode;
321
322     baru->data = dataBaru;
323     baru->next = bantu;
324
325     sebelum->next = baru;
326
327     if(bantu == head) {
328         head = baru;
329     }
330
331     cout<<"Data  \<<dataBaru<<"\<<endl;
332     cout<<"sebelum data \<<nextData<<"\<<endl;
333     } else {
334         cout<<"Tidak terdapat data \<<nextData<<"\<<endl;
335         Linked List."<<endl;
336     }
337
338 void sisipkanSetelah() {
339     if(isEmpty() == 0) {
340         TNode *bantu;
341         string prevData;
342         bool apaAda;
343

```

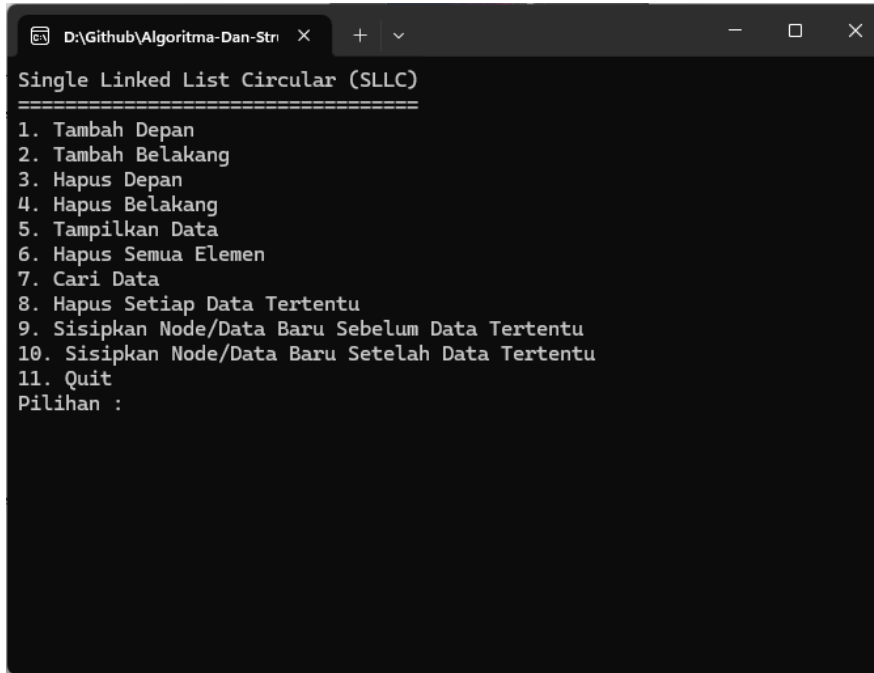
```

344     bantu = head;
345
346     cout<<"Sisipkan data baru setelah data : ";
347     cin>>prevData;
348
349     do {
350         if(prevData == bantu->data) {
351             apaAda = true;
352             break;
353         } else {
354             bantu = bantu->next;
355         }
356     } while(bantu != head);
357
358     if(apaAda == true) {
359         cout<<"Masukan data yang ditambahkan : ";
360         cin>>dataBaru;
361
362         TNode *baru;
363         baru = new TNode;
364
365         baru->data = dataBaru;
366         baru->next = bantu->next;
367
368         bantu->next = baru;
369
370         if(bantu == tail) {
371             tail = baru;
372         }
373
374         cout<<"Data  \<<dataBaru<<"\<<endl;
375     } else {
376         cout<<"Data  \<<prevData<<"\<<endl;
377     }

```

```
378. } else cout<<"Tidak terdapat data pada Linked List."<<endl;
```

## B. Output Program



The screenshot shows a terminal window with the title "D:\Github\Algoritma-Dan-Str". The output text is as follows:

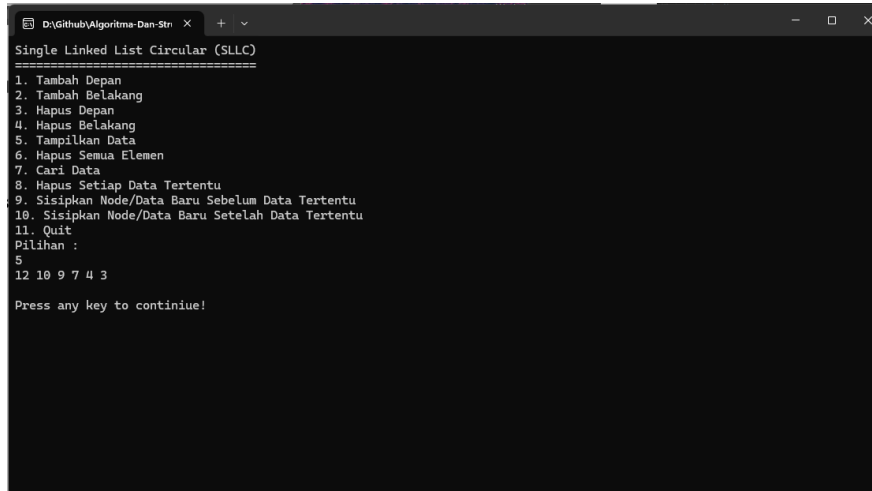
```
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua Elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Quit
Pilihan :
```

*Gambar 14. Modul 3 Output Saol 1*

## SOAL 2

Lakukan tambah data depan 3, 4, 7, 9, 10, 12 dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

### A. Output Program



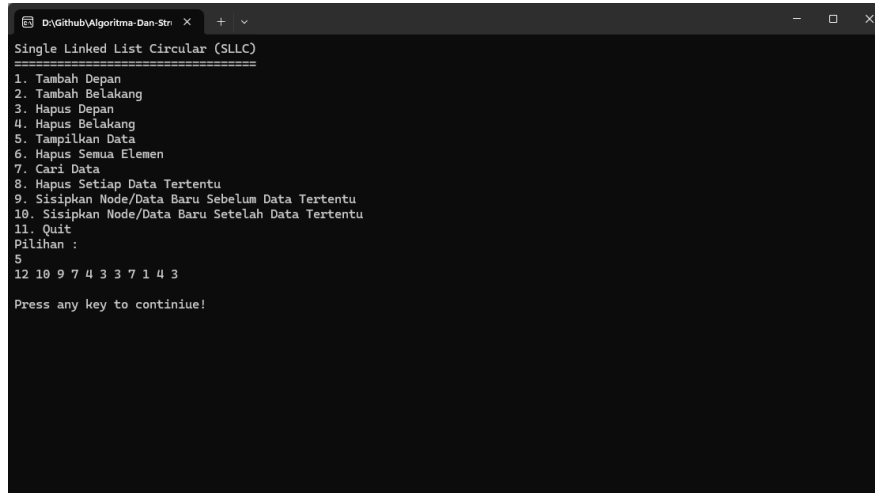
```
D:\Github\Algoritma-Dan-Str x + v
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua Elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Quit
Pilihan :
5
12 10 9 7 4 3
Press any key to continue!
```

*Gambar 15. Modul 3 Screenshot Output Saol 2*

### SOAL 3

Lakukan tambah data belakang 3, 7, 1, 4, 3 dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

#### A. Output Program



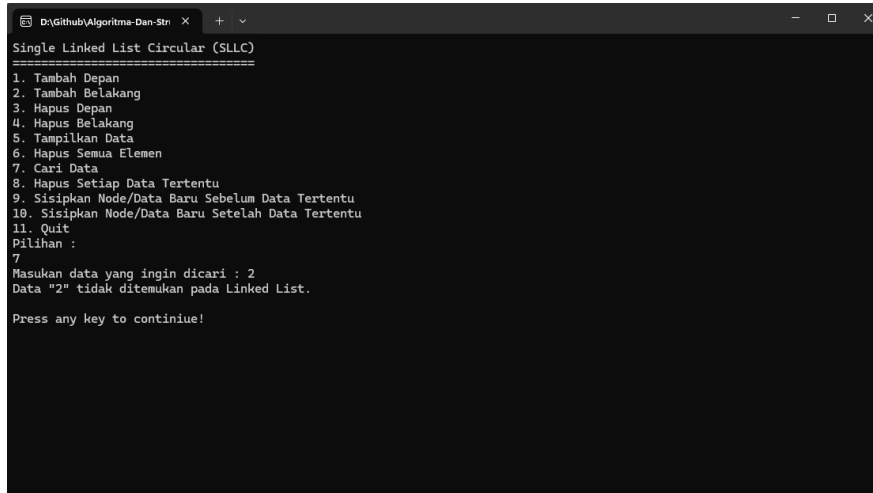
```
D:\Github\Algoritma-Dan-Str > + v
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua Elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Quit
Pilihan :
5
12 10 9 7 4 3 3 7 1 4 3
Press any key to continue!
```

Gambar 16. Modul 3 Screenshot Output Soal 3

## SOAL 4

Apa yang terjadi jika mencari angka 2 pada Single Linked List Circular (SLLC) pada data yang telah ditambahkan/dimasukkan sebelumnya dan screenshoot hasilnya !

### A. Output Program



```
D:\Github\Algoritma-Dan-Str > + v
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua Elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Quit
Pilihan :
7
Masukan data yang ingin dicari : 2
Data "2" tidak ditemukan pada Linked List.
Press any key to continue!
```

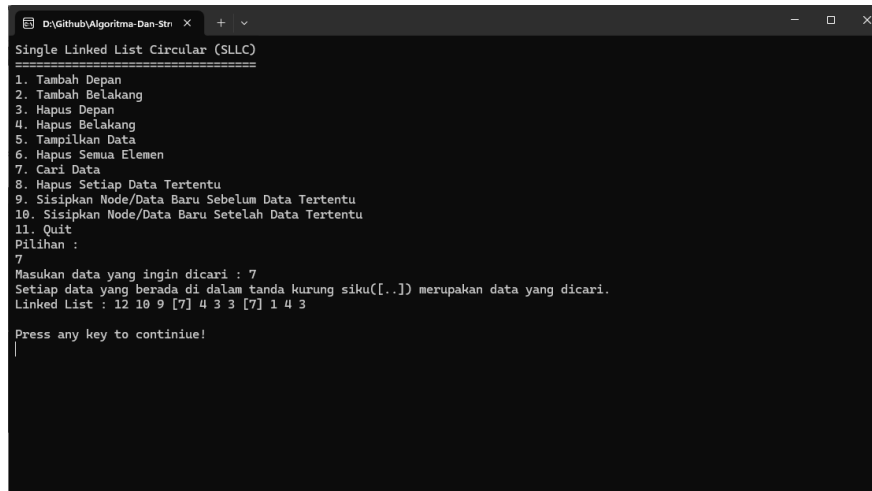
Gambar 17. Modul 3 Screenshot Output Soal 4



## SOAL 5

Coba cari angka 7 dan screenshoot hasilnya !

### A. Output Program



```
D:\Github\Algoritma-Dan-Str x + - x
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua Elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Quit
Pilihan :
7
Masukan data yang ingin dicari : 7
Setiap data yang berada di dalam tanda kurung siku([..]) merupakan data yang dicari.
Linked List : 12 10 9 [7] 4 3 3 [7] 1 4 3

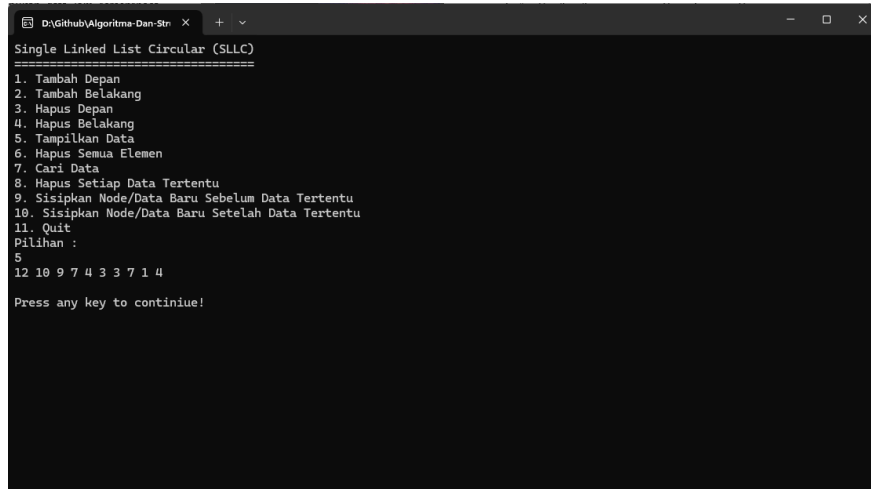
Press any key to continue!
|
```

*Gambar 18. Modul 3 Output Soal 5*

## SOAL 6

Lakukan hapus belakang dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

### A. Output Program



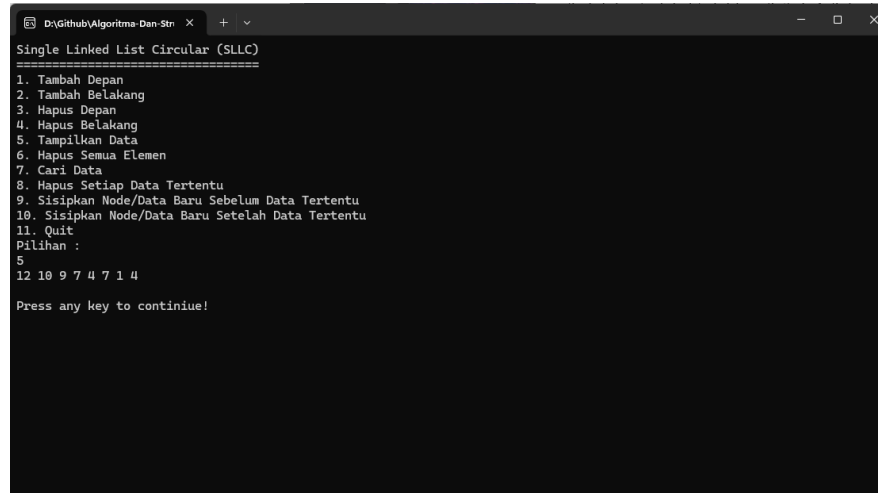
```
D:\Github\Algoritma-Dan-Str x + v
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua Elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Quit
Pilihan :
5
12 10 9 7 4 3 3 7 1 4
Press any key to continue!
```

*Gambar 19. Modul 3 Output Soal 6*

## SOAL 7

Lakukan hapus setiap angka 3 dan kemudian lakukan tampilkan data lalu screenshoot hasilnya !

### A. Output Program



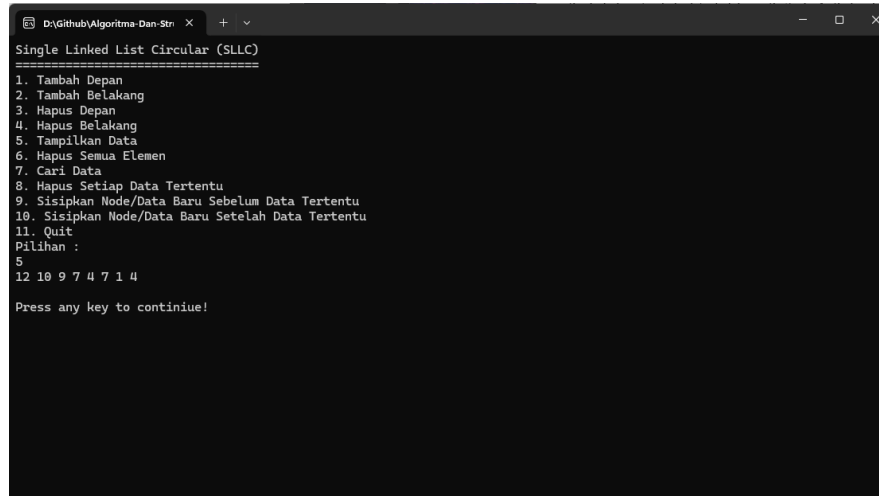
```
D:\Github\Algoritma-Dan-Str >
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua Elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Quit
Pilihan :
5
12 10 9 7 4 7 1 4
Press any key to continue!
```

Gambar 20. Modul 3 Output Soal 7

## SOAL 8

Tampilkan data lalu jelaskan yang mana head dan yang mana tail.

### A. Output Program



```
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua Elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Quit
Pilihan :
5
12 10 9 7 4 7 1 4
Press any key to continue!
```

Gambar 21. Modul 3 Output Saol 8

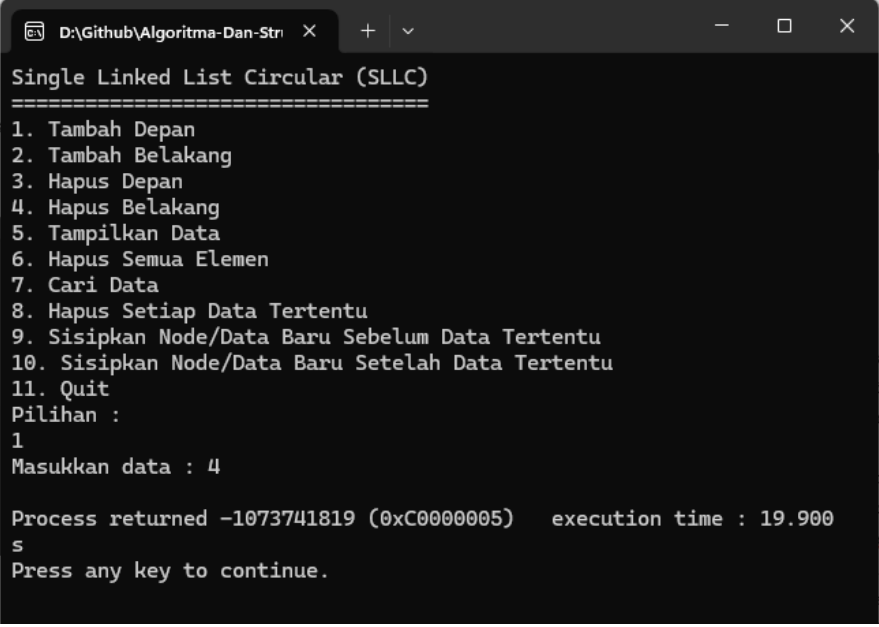
### B. Pembahasan

Berdasarkan dari gambar 8 urutan data yang ditampilkan secara berurutan adalah “12, 10, 9, 7, 4, 7, 1, 4”. Maka data yang menjadi head adalah data yang berada paling kiri yaitu 12, karena head adalah node yang menunjuk ke data paling pertama yang diakses dalam list. Sedangkan data yang menjadi tail adalah data yang berada paling kanan yaitu 4, karena tail adalah node terakhir dalam list, yang pointer next-nya menunjuk kembali ke head.

## SOAL 9

Jika baris ke 103 dan 104 dihapus maka apa yang akan terjadi pada saat memasukkan data, dan jelaskan mengapa?

### A. Output Program



```
D:\Github\Algoritma-Dan-Stri X + v - □ X
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua Elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Quit
Pilihan :
1
Masukkan data : 4

Process returned -1073741819 (0xC0000005)   execution time : 19.900
s
Press any key to continue.
```

Gambar 22. Modul 3 Output Soal 9

### B. Pembahasan

Di awal kita masih bisa menjalankan programnya, tetapi kita mencoba untuk menambahkan data menggunakan opsi menu satu dan dua maka program mungkin akan berhenti seketika. Ini disebabkan baris yang dihapus merupakan isi dari fungsi isEmpty(), yang mana fungsi ini pada normalnya akan mengembalikan value berupa 1 atau 0. Namun ketika pada kondisi di mana isi dari fungsi dari isEmpty() dihapus, nilai yang akan dikembalikan adalah garbage value atau nilai tidak pasti bisa saja 1 ataupun -11. Padahal nilai dari fungsi isEmpty() digunakan untuk logika pada fungsi penambahan data baru maupun fungsi lainnya. Program bisa salah mengira bahwa list sudah berisi, dan memproses else membuat struktur pointer jadi salah dan berpotensi crash.

## SOAL 10

Jelaskan apa itu variabel head dan tail pada sllc!

### A. Pembahasan

Variabel head adalah pointer yang menunjuk ke simpul (node) pertama dalam struktur Single Linked List Circular (SLLC. Ketika pengguna ingin menampilkan seluruh isi list, maka penelusuran dimulai dari node yang ditunjuk oleh head. Karena sifat circular dari list ini, proses penelusuran akan berakhir kembali ke node head setelah semua node dilalui. Oleh karena itu, head memegang peran krusial sebagai acuan awal dan penanda batas dalam proses traversal list.

Variabel tail adalah pointer yang menunjuk ke simpul terakhir dalam SLLC. Tidak seperti linked list biasa yang node terakhirnya menunjuk ke NULL, pada SLLC node terakhir justru menunjuk kembali ke node pertama (yaitu head). Dengan demikian, tail->next akan selalu menunjuk ke head, yang menjadikan struktur data ini berbentuk melingkar atau circular. tail juga sering digunakan saat menambahkan node di bagian belakang list, karena node baru akan ditempatkan setelah node yang ditunjuk oleh tail, lalu tail diperbarui agar menunjuk ke node yang baru ditambahkan.

## MODUL 4 : DOUBLE LINK LIST

### SOAL 1

Lengkapi coding pada function tambahDepanH() agar bisa berjalan dengan lancar. running, simpan program, dan screenshoot hasil running !

```
1  #include <conio.h>
2  #include <iostream>
3  #include <stdlib.h>
4
5  using namespace std;
6
7  typedef struct TNode {
8      string data;
9      TNode *next;
10     TNode *prev;
11 };
12
13 TNode *head, *tail;
14
15 int pil, menu;
16 char pilihan[1];
17 string dataDaru;
18
19 void inith();
20 void inihT();
21 int isEmpty();
22 int isEmptyT();
23
24 void tambahDepanH();
25 void tambahDepanHT();
26 void tambahBelakangH();
27 void tambahBelakangHT();
28 void hapusDepanH();
29 void hapusDepanHT();
30 void hapusBelakangH();
31 void hapusBelakangHT();
32 void tampilkanH();
33 void tampilkanHT();
34 void clrH();
35 void clrHT();
36
37 int main()
38 {
39     menu:
40     cout<<"Double Linked List Non Circular (DLNC)"<<endl;
41     cout<<"-----"<<endl;
42     cout<<"Silahkan pilih program DLNC yang ingin dijalankan!"<<endl;
43     cout<<"1. DLNC dengan Head"<<endl;
44     cout<<"2. DLNC dengan Head dan Tail"<<endl;
45     cout<<"3. Quit"<<endl;
46     cout<<"Pilihan : ";
47     cin>>menu;
48     system("cls");
49     if(menu==1){
50         do {
51             cout<<"Double Linked List Non Circular (DLNC) (Head)"<<endl;
52             cout<<"-----"<<endl;
53             cout<<"1. Tambah Depan"<<endl;
54             cout<<"2. Tambah Belakang"<<endl;
55             cout<<"3. Tampilkan Data"<<endl;
56             cout<<"4. Hapus Depan"<<endl;
57             cout<<"5. Hapus Belakang"<<endl;
58             cout<<"6. Reset"<<endl;
59             cout<<"7. Kembali ke Menu"<<endl;
60             cout<<"Pilihan : ";
61             cin>>pilihan;
62             pil=atoi(pilihan);
```

```

63
64     switch(pil) {
65     case 1:
66         tambahDepanH();
67         break;
68     case 2:
69         tambahBelakangH();
70         break;
71     case 3:
72         tampilkanH();
73         break;
74     case 4:
75         hapusDepanH();
76         break;
77     case 5:
78         hapusBelakangH();
79         break;
80     case 6:
81         clearH();
82         break;
83     default:
84         system("cls");
85         goto menu;
86     }
87
88     cout<<"\npress any key to continue"<<endl;
89     getch();
90     system("cls");
91
92 } while (pil<7);
93 } else if(menu==2){
94     do {
95         cout<<"Double Linked List Non Circular (DLNLC) (Head dan Tail)"<<endl;
96         cout<<"===== "<<endl;
97         cout<<"1. Tambah Depan"<<endl;
98         cout<<"2. Tambah Belakang"<<endl;
99         cout<<"3. Tampilkan Data"<<endl;
100        cout<<"4. Hapus Depan"<<endl;
101        cout<<"5. Hapus Belakang"<<endl;
102        cout<<"6. Reset"<<endl;
103        cout<<"7. Kembali ke Menu"<<endl;
104        cout<<"Pilihan : ";
105        cin>>pilihan;
106        pil=atoi(pilihan);
107
108        switch(pil) {
109        case 1:
110            tambahDepanHT();
111            break;
112        case 2:
113            tambahBelakangHT();
114            break;
115        case 3:
116            tampilkanHT();
117            break;
118        case 4:
119            hapusDepanHT();
120            break;
121        case 5:
122            hapusBelakangHT();
123            break;
124        case 6:
125            clearHT();
126            break;
127        default:
128            system("cls");
129            goto menu;
130        }
131
132        cout<<"\npress any key to continue"<<endl;
133        getch();
134        system("cls");
135
136    } while (pil<7);
137 } else {
138     cout<<"\nBERKATA KASIH"<<endl;
139     cout<<"Program was made by Nana (NIM)."<<endl;
140 }
141 }
142

```



```

143 void initH(){
144     head = NULL;
145 }
146
147 void initHT(){
148     head = NULL;
149     tail = NULL;
150 }
151
152 int isEmptyH(){
153     if(head == NULL) return 1;
154     else return 0;
155 }
156
157 int isEmptyHT(){
158     if(tail == NULL) return 1;
159     else return 0;
160 }
161
162 void tambahDepanH() {
163     cout<<"Masukkan data : ";
164
165
166
167
168
169
170
171
172
173
174
175
176
177     cout << "Data \t"<<dataBaru<<"\n" berhasil dimasukkan di bagian depan.";
178 }
179
180 void tambahDepanHT() {
181     cout<<"Masukkan data : ";
182     cin>>dataBaru;
183     TNode *baru;
184     baru = new TNode;
185     baru->data = dataBaru;
186     baru->next = NULL;
187     baru->prev = NULL;
188     if(isEmptyHT() == 1) {
189         head = baru;
190         tail = baru;
191     } else {
192         baru->next = head;
193         head->prev = baru;
194         head = baru;
195     }
196     cout << "Data \t"<<dataBaru<<"\n" berhasil dimasukkan di bagian depan.";
197 }
198
199 void tambahBelakangH() {
200     cout<<"Masukkan data : ";
201     cin>>dataBaru;
202     TNode *baru, *bantu;
203     baru = new TNode;
204     baru->data = dataBaru;
205     baru->next = NULL;
206     baru->prev = NULL;
207     if(isEmptyH() == 1) {
208         head = baru;
209     } else {
210         bantu = head;
211         while(bantu->next != NULL){
212             bantu = bantu->next;
213         }
214         bantu->next = baru;
215         baru->prev = bantu;
216     }
217     cout << "Data \t"<<dataBaru<<"\n" berhasil dimasukkan di bagian belakang.";
218 }
219

```

```

220 void tambahBelakangHT() {
221     cout<<"Masukkan data : ";
222     cin>>dataBaru;
223     TNode *baru;
224     baru = new TNode;
225     baru->data = dataBaru;
226     baru->next = NULL;
227     baru->prev = NULL;
228     if(isEmptyHT() == 1) {
229         head = baru;
230         tail = baru;
231     } else {
232         tail->next = baru;
233         baru->prev = tail;
234         tail = baru;
235     }
236     cout << "Data \""<<dataBaru<<"\" berhasil dimasukkan di bagian belakang.";
237 }
238
239 void tampilkanH() {
240     TNode *bantu;
241     bantu = head;
242     if(isEmptyHT() == 0) {
243         while(bantu != NULL) {
244             cout<<bantu->data<<" ";
245             bantu = bantu->next;
246         }
247         cout<<endl;
248     } else cout<<"Tidak terdapat data pada Linked List";
249 }
250
251 void tampilkanHT() {
252     TNode *bantu;
253     bantu = head;
254     if(isEmptyHT() == 0) {
255         while(bantu != tail->next) {
256             cout<<bantu->data<<" ";
257             bantu = bantu->next;
258         }
259         cout<<endl;
260     } else cout<<"Tidak terdapat data pada Linked List";
261 }
262
263 void hapusDepanH() {
264     TNode *hapus;
265     string data;
266     if(isEmptyHT() == 0) {
267         hapus = head;
268         data = hapus->data;
269         if(head->next != NULL) {
270             head = head->next;
271             head->prev = NULL;
272         } else {
273             inith();
274         }
275         delete hapus;
276         cout<<"Data \""<<data<<"\" yang berada di depan telah berhasil dihapus.";
277     } else cout<<"Tidak terdapat data pada Linked List";
278 }
279
280 void hapusDepanHT() {
281     TNode *hapus;
282     string data;
283     if(isEmptyHT() == 0) {
284         hapus = head;
285         data = hapus->data;

```

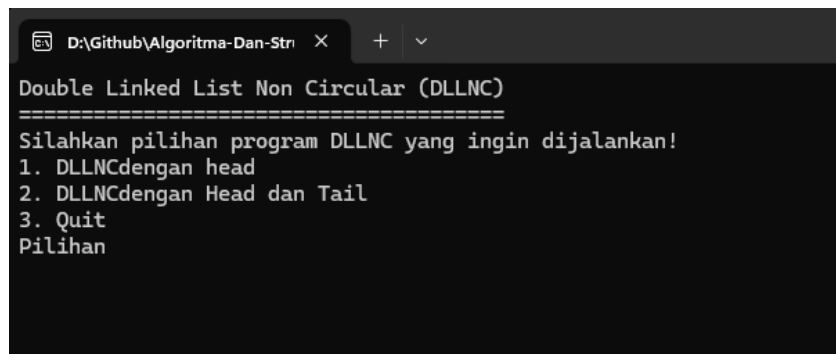
```

286         if(head->next != NULL) {
287             head = head->next;
288             head->prev = NULL;
289         } else {
290             initHT();
291         }
292         delete hapus;
293         cout<<"Data \"<<data<<\" yang berada di depan telah berhasil dihapus.";
294     } else cout<<"Tidak terdapat data pada Linked List";
295 }
296
297 void hapusBelakangH() {
298     TNode *hapus;
299     string data;
300     if(isEmptyH() == 0) {
301         hapus = head;
302         while(hapus->next != NULL){
303             hapus = hapus->next;
304         }
305         data = hapus->data;
306         if(head->next != NULL) {
307             hapus->prev->next = NULL;
308         } else {
309             initH();
310         }
311         delete hapus;
312         cout<<"Data \"<<data<<\" yang berada di belakang telah berhasil dihapus.";
313     } else cout<<"Tidak terdapat data pada Linked List";
314 }
315
316 void hapusBelakangHT() {
317     TNode *hapus;
318     string data;
319     if(isEmptyHT() == 0) {
320         hapus = tail;
321         data = hapus->data;
322         if(head->next != NULL) {
323             tail = tail->prev;
324             tail->next = NULL;
325         } else {
326             initHT();
327         }
328         delete hapus;
329         cout<<"Data \"<<data<<\" yang berada di belakang telah berhasil dihapus.";
330     } else cout<<"Tidak terdapat data pada Linked List";
331 }
332
333 void clearH() {
334     TNode *bantu, *hapus;
335     bantu = head;
336     while(bantu != NULL) {
337         hapus = bantu;
338         bantu = bantu->next;
339         delete hapus;
340     }
341     initH();
342     cout<<"Seluruh data pada Linked List telah dibersihkan.";
343 }
344
345 void clearHT() {
346     TNode *bantu, *hapus;
347     bantu = head;
348     while(bantu != NULL) {
349         hapus = bantu;
350         bantu = bantu->next;
351         delete hapus;
352     }
353     initHT();
354     cout<<"Seluruh data pada Linked List telah dibersihkan.";
355 }

```

Gambar 23. Modul 4 Saol 1

## A. Screenshoot



```
D:\Github\Algoritma-Dan-Stri X + v
Double Linked List Non Circular (DLLNC)
=====
Silahkan pilihan program DLLNC yang ingin dijalankan!
1. DLLNCdengan head
2. DLLNCdengan Head dan Tail
3. Quit
Pilihan
```

*Gambar 24. Modul 4 Output Saol 1*

## SOAL 2

Apa fungsi next pada coding?

### A. Pembahasan

Dalam kode program Double Linked List Non Circular (DLLNC), atribut next berfungsi untuk menunjukkan simpul (node) berikutnya dari sebuah node dalam struktur linked list. Dengan kata lain, next adalah pointer yang menyimpan alamat node selanjutnya, memungkinkan traversal (penelusuran) ke depan. Dan setiap node pasti memiliki pointer next.

### SOAL 3

Apa fungsi prev pada coding?

#### A. Pembahasan

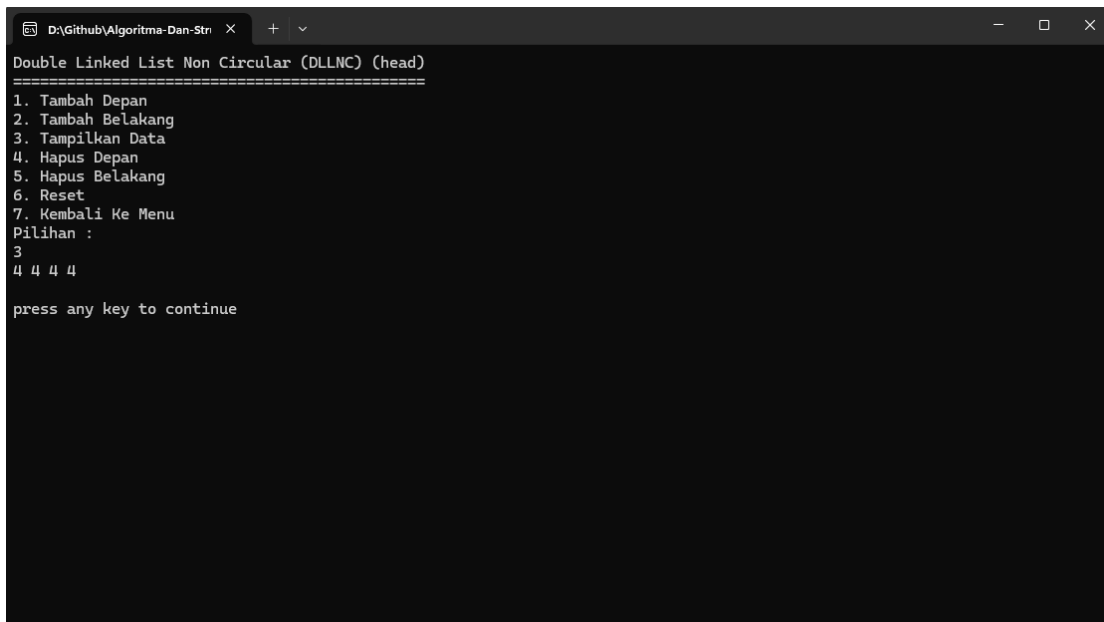
Dalam kode program Double Linked List Non Circular (DLLNC), atribut prev berfungsi untuk menunjuk ke simpul (node) sebelumnya dari sebuah node dalam linked list. Dengan kata lain, prev adalah pointer yang menyimpan alamat node sebelumnya, memungkinkan traversal (penelusuran) ke belakang. Dan setiap node pasti memiliki pointer prev.

## SOAL 4

Gantilah baris 244 dan 256 dari `cout<<bantu->data<<' '`; menjadi `cout<<head->data<<' '`; lalu jawab pertanyaan berikut :

- A. Apa yang terjadi jika anda menambahkan beberapa data pada program lalu tampilkan datanya, dan screenshoot hasilnya.
- B. Jelaskan mengapa hal tersebut bisa terjadi dan data apa yang ditampilkan oleh program?

### A. Pembahasan



```
D:\Github\Algoritma-Dan-Stru >
Double Linked List Non Circular (DLLNC) (head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
3
4 4 4 4
press any key to continue
```

Gambar 25. Modul 4. Output Perubahan baris 244 dan 256 Soal 4

Ketika saya menambahkan data depan secara berurutan yaitu 1, 2, 3, 4. Lalu saya menjalankan fungsi tampilkan data maka output yang ditampilkan adalah 4 4 4 4. Hal tersebut terjadi karena, perubahan kode pada baris 244 dan 256 membuat fungsi menampilkan data akan selalu menampilkan data dari node head (`head->data`), meskipun pointer bantu bergerak sepanjang linked list. Jika saya memasukkan data 1, 2, 3, 4 secara berurutan menggunakan DLLNC (head) atau DLLNC (head dan tail), linked list yang akan tersusun adalah 4, 3, 2, 1. Dalam struktur linked list, head selalu menunjuk ke node pertama yang berisi data “4”. Dalam fungsi `tampilkanH()` atau

tampilkanHT(), loop while akan berjalan sebanyak jumlah node(4 kali) karena bantu masih bergerak sepanjang linked list. Namun pada setiap iterasi, yang dicetak adalah head->data yang selalu berisi data"4". Akibatnya, output yang muncul adalah "4" sebanyak node dalam linked list (4 kali).

Jadi dengan mengubah code pada baris 244 dan 256 , program tidak lagi mencetak seluruh data dalam linked list, melainkan hanya mencetak data pada node head sebanyak jumlah node yang ada. Fungsi traversal pada fungsi tampilkan menjadi tidak berguna karena yang ditampilkan hanya elemen pertama saja secara berulang.



## MODUL 5 : SORTING

### SOAL 1

Buat Program Sederhana Menggunakan Nama dan Angka NIM Masing-masing :

- Insertion Sort (Nama)
- Merge Sort (Nama)
- Shell Sort (Nama)
- Quick Sort (NIM)
- Bubble Sort (NIM)
- Selection Sort (NIM)



*Gambar 26. Modul 5 Soal 1*

## A. Source Code

Tabel 5. Modul 5 Source Code Soal 1

1	#include <iostream>
2	#include <functional>
3	#include <chrono>
4	#include <string>
5	#include <iomanip>
6	#include <conio.h>
7	
8	using namespace std;
9	
10	// Ganti dengan nama dan NIM Anda
11	string name = "YourName"; // Ganti dengan nama Anda
12	string id = "YourNIM"; // Ganti dengan NIM Anda
13	
14	// Function untuk mengukur waktu eksekusi
15	void timeSort(const function<void()>& sortFunc, const string& sortName) {
16	auto start = chrono::high_resolution_clock::now();
17	sortFunc();
18	auto end = chrono::high_resolution_clock::now();
19	chrono::duration<double> duration = end - start;
20	
21	cout << fixed << setprecision(10);
22	cout << sortName << " took " << duration.count() << " seconds\n";
23	}
24	
25	// 1. INSERTION SORT untuk karakter dalam string
26	void insertionSort(string &str) {
27	for (int i = 1; i < str.size(); i++) {
28	char key = str[i];
29	int j = i - 1;
30	
31	while (j >= 0 && str[j] > key) {
32	str[j + 1] = str[j];

33	j--;
34	}
35	
36	str[j + 1] = key;
37	}
38	}
39	
40	// 2. MERGE SORT untuk karakter dalam string
41	void merge(string &str, int left, int mid, int right) {
42	int n1 = mid - left + 1;
43	int n2 = right - mid;
44	
45	char *tempL = new char[n1];
46	char *tempR = new char[n2];
47	
48	for (int i = 0; i < n1; i++) tempL[i] = str[left + i];
49	for (int j = 0; j < n2; j++) tempR[j] = str[mid + 1 + j];
50	
51	int i = 0, j = 0, k = left;
52	
53	while (i < n1 && j < n2) {
54	if (tempL[i] <= tempR[j]) {
55	str[k] = tempL[i];
56	i++;
57	} else {
58	str[k] = tempR[j];
59	j++;
60	}
61	k++;
62	}
63	
64	while (i < n1) {
65	str[k] = tempL[i];
66	i++;
67	k++;
68	}

```

69
70     while (j < n2) {
71         str[k] = tempR[j];
72         j++;
73         k++;
74     }
75
76     delete[] tempL;
77         delete[] tempR;
78 }
79
80 void mergeSort(string &str, int left, int right) {
81     if (left < right) {
82         int mid = left + (right - left) / 2;
83         mergeSort(str, left, mid);
84         mergeSort(str, mid + 1, right);
85         merge(str, left, mid, right);
86     }
87 }
88
89 // 3. SHELL SORT untuk karakter dalam string
90 void shellSort(string &str, int n) {
91     for (int gap = n/2; gap > 0; gap /= 2) {
92         for (int i = gap; i < n; i++) {
93             char temp = str[i];
94
95             int j;
96             for (j = i; j >= gap && str[j - gap] > temp; j -=
gap) {
97                 str[j] = str[j - gap];
98             }
99
100             str[j] = temp;
101         }
102     }
103 }

```

104	
105	// 4. BUBBLE SORT untuk karakter dalam string
106	void bubbleSort(string &str) {
107	for (int i = 0; i < str.size() - 1; i++) {
108	bool swapped = false;
109	
110	for (int j = 0; j < str.size() - i - 1; j++) {
111	if (str[j] > str[j + 1]) {
112	swap(str[j], str[j + 1]);
113	swapped = true;
114	}
115	}
116	
117	if (!swapped) break;
118	}
119	}
120	
121	// 5. QUICK SORT untuk karakter dalam string
122	int partition(string &str, int low, int high) {
123	char pivot = str[high];
124	int i = (low - 1);
125	
126	for (int j = low; j <= high - 1; j++) {
127	if (str[j] <= pivot) {
128	i++;
129	swap(str[i], str[j]);
130	}
131	}
132	
133	swap(str[i + 1], str[high]);
134	return (i + 1);
135	}
136	
137	void quickSort(string &str, int low, int high) {
138	if (low < high) {

```

139         int p_idx = partition(str, low, high);
140         quickSort(str, low, p_idx - 1);
141         quickSort(str, p_idx + 1, high);
142     }
143 }
144
145 // 6. SELECTION SORT untuk karakter dalam string
146 void selectionSort(string &str) {
147     for (int i = 0; i < str.size() - 1; i++) {
148         int minIndex = i;
149
150         for (int j = i + 1; j < str.size(); j++) {
151             if (str[j] < str[minIndex]) {
152                 minIndex = j;
153             }
154         }
155
156         swap(str[i], str[minIndex]);
157     }
158 }
159
160 int main() {
161     int ch;
162     string temp;
163
164     do {
165         cout << "+=====+" << endl;
166         cout << "| Sorting Algorithm |" << endl;
167         cout << "+=====+" << endl;
168         cout << "| 1. Insertion Sort |" << endl;
169         cout << "| 2. Merge Sort      |" << endl;
170         cout << "| 3. Shell Sort      |" << endl;
171         cout << "| 4. Bubble Sort     |" << endl;
172         cout << "| 5. Quick Sort      |" << endl;
173         cout << "| 6. Selection Sort  |" << endl;
174         cout << "| 7. Exit           |" << endl;

```

```

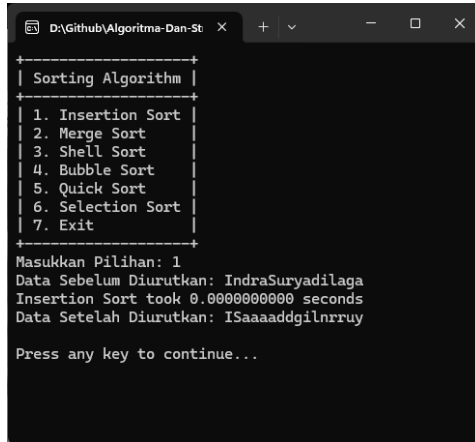
175         cout << "+=====+" << endl;
176         cout << "Masukkan Pilihan: ";
177         cin >> ch;
178
179         switch(ch) {
180             case 1:
181                 temp = name;
182                 cout << "Data Sebelum Diurutkan: " << temp <<
endl;
183                 timeSort([&]() {insertionSort(temp); },
"Insertion Sort");
184                 cout << "Data Setelah Diurutkan: " << temp <<
endl;
185                 break;
186             case 2:
187                 temp = name;
188                 cout << "Data Sebelum Diurutkan: " << temp <<
endl;
189                 timeSort([&]() {mergeSort(temp, 0, temp.size() -
1); }, "Merge Sort");
190                 cout << "Data Setelah Diurutkan: " << temp <<
endl;
191                 break;
192             case 3:
193                 temp = name;
194                 cout << "Data Sebelum Diurutkan: " << temp <<
endl;
195                 timeSort([&]() {shellSort(temp, temp.size()); },
"Shell Sort");
196                 cout << "Data Setelah Diurutkan: " << temp <<
endl;
197                 break;
198             case 4:
199                 temp = id;
200                 cout << "Data Sebelum Diurutkan: " << temp <<
endl;
201                 timeSort([&]() {bubbleSort(temp); }, "Bubble
Sort");
202                 cout << "Data Setelah Diurutkan: " << temp <<
endl;
203                 break;
204             case 5:
205                 temp = id;

```

206	cout << "Data Sebelum Diurutkan: " << temp <<
207	endl;
207	timeSort([&]() {quickSort(temp, 0, temp.size() -
208	1); }, "Quick Sort");
208	cout << "Data Setelah Diurutkan: " << temp <<
209	endl;
209	break;
210	case 6:
211	temp = id;
212	cout << "Data Sebelum Diurutkan: " << temp <<
213	endl;
213	timeSort([&]() {selectionSort(temp); },
214	"Selection Sort");
214	cout << "Data Setelah Diurutkan: " << temp <<
215	endl;
215	break;
216	case 7:
217	cout << "Terima Kasih" << endl;
218	cout << "This Program Was Made by [Indra
219	Suryadilaga] (2410817310014)" << endl;
220	break;
220	default:
221	cout << "Opsi Tidak Valid. Silahkan Coba Lagi."
222	<< endl;
222	}
223	cout << "\nPress any key to continue..." << endl;
224	getch();
225	system("cls");
226	} while (ch != 7);
227	
228	return 0;
229	}



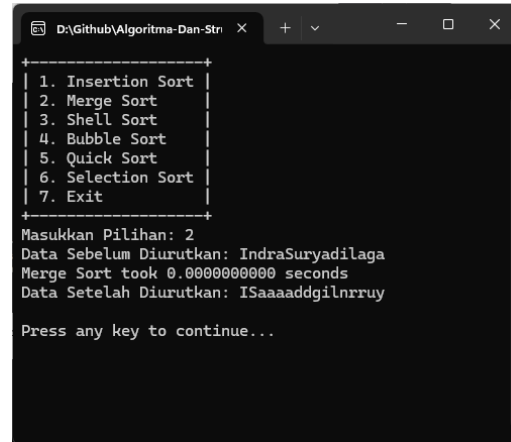
## A. Output Program



```
D:\Github\Algoritma-Dan-Str x + - □ x
+-----+
| Sorting Algorithm |
+-----+
| 1. Insertion Sort |
| 2. Merge Sort    |
| 3. Shell Sort    |
| 4. Bubble Sort   |
| 5. Quick Sort    |
| 6. Selection Sort|
| 7. Exit          |
+-----+
Masukkan Pilihan: 1
Data Sebelum Diurutkan: IndraSuryadilaga
Insertion Sort took 0.000000000 seconds
Data Setelah Diurutkan: ISaaaaddgilnrruy

Press any key to continue...
```

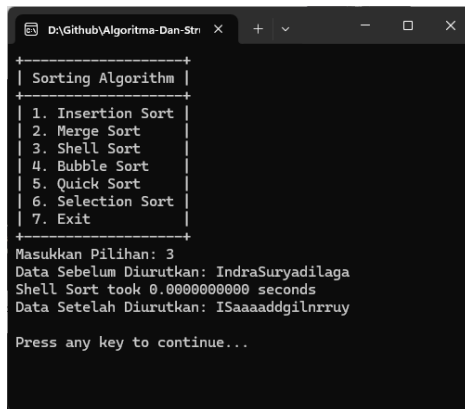
Gambar 27. Modul 5 Output Insertion Sort Saol 1



```
D:\Github\Algoritma-Dan-Str x + - □ x
+-----+
| 1. Insertion Sort |
| 2. Merge Sort    |
| 3. Shell Sort    |
| 4. Bubble Sort   |
| 5. Quick Sort    |
| 6. Selection Sort|
| 7. Exit          |
+-----+
Masukkan Pilihan: 2
Data Sebelum Diurutkan: IndraSuryadilaga
Merge Sort took 0.000000000 seconds
Data Setelah Diurutkan: ISaaaaddgilnrruy

Press any key to continue...
```

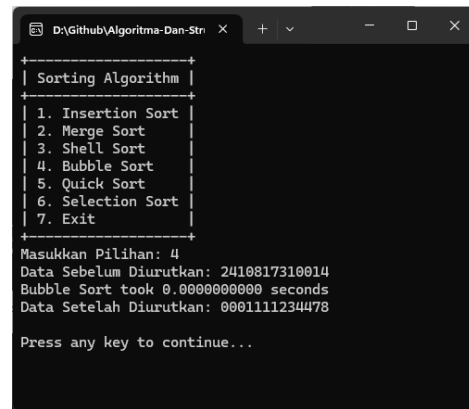
Gambar 28. Modul 5 Output Merge Sort Saol 1



```
D:\Github\Algoritma-Dan-Str x + - □ x
+-----+
| Sorting Algorithm |
+-----+
| 1. Insertion Sort |
| 2. Merge Sort    |
| 3. Shell Sort    |
| 4. Bubble Sort   |
| 5. Quick Sort    |
| 6. Selection Sort|
| 7. Exit          |
+-----+
Masukkan Pilihan: 3
Data Sebelum Diurutkan: IndraSuryadilaga
Shell Sort took 0.000000000 seconds
Data Setelah Diurutkan: ISaaaaddgilnrruy

Press any key to continue...
```

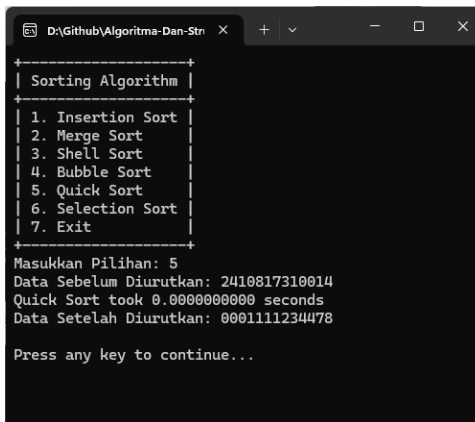
Gambar 29. Modul 5 Output Shell Sort Saol 1



```
D:\Github\Algoritma-Dan-Str x + - □ x
+-----+
| Sorting Algorithm |
+-----+
| 1. Insertion Sort |
| 2. Merge Sort    |
| 3. Shell Sort    |
| 4. Bubble Sort   |
| 5. Quick Sort    |
| 6. Selection Sort|
| 7. Exit          |
+-----+
Masukkan Pilihan: 4
Data Sebelum Diurutkan: 2410817310014
Bubble Sort took 0.000000000 seconds
Data Setelah Diurutkan: 0001111234478

Press any key to continue...
```

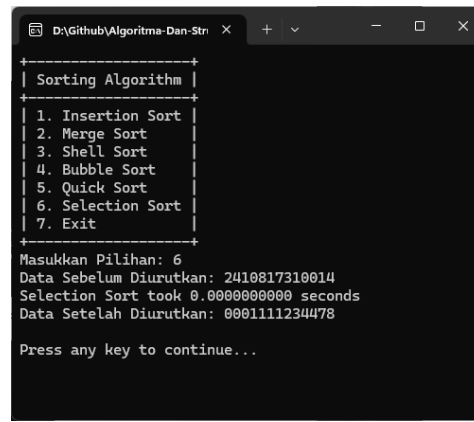
Gambar 30. Modul 5 Output Bubble Sort Saol 1



```
D:\Github\Algoritma-Dan-Str x + - □ x
+-----+
| Sorting Algorithm |
+-----+
| 1. Insertion Sort |
| 2. Merge Sort    |
| 3. Shell Sort    |
| 4. Bubble Sort   |
| 5. Quick Sort    |
| 6. Selection Sort|
| 7. Exit          |
+-----+
Masukkan Pilihan: 5
Data Sebelum Diurutkan: 2410817310014
Quick Sort took 0.000000000 seconds
Data Setelah Diurutkan: 0001111234478

Press any key to continue...
```

Gambar 31. Modul 5 Output Quick Sort Saol 1



```
D:\Github\Algoritma-Dan-Str x + - □ x
+-----+
| Sorting Algorithm |
+-----+
| 1. Insertion Sort |
| 2. Merge Sort    |
| 3. Shell Sort    |
| 4. Bubble Sort   |
| 5. Quick Sort    |
| 6. Selection Sort|
| 7. Exit          |
+-----+
Masukkan Pilihan: 6
Data Sebelum Diurutkan: 2410817310014
Selection Sort took 0.000000000 seconds
Data Setelah Diurutkan: 0001111234478

Press any key to continue...
```

Gambar 32. Modul 5 Output Selection Sort Saol 1

## **B. Pembahasan**

### **Alur Program Sorting**

Ketika program dijalankan, sistem akan menampilkan menu utama dengan 7 pilihan sorting algorithm yang tersedia. Program menggunakan struktur do-while loop yang akan terus berjalan hingga user memilih opsi 7 (Exit). Setiap kali user memilih opsi 1-6, program akan melakukan proses sorting sesuai dengan algoritma yang dipilih.

Untuk opsi 1-3 (Insertion Sort, Merge Sort, dan Shell Sort), program akan menggunakan string name yang berisi "IndraSuryadilaga" sebagai data input. Sedangkan untuk opsi 4-6 (Bubble Sort, Quick Sort, dan Selection Sort), program menggunakan string id yang berisi "2410817310014" sebagai data input.

Dalam setiap proses sorting, program akan menampilkan data sebelum diurutkan, kemudian memanggil fungsi `timeSort()` yang menggunakan lambda function untuk mengukur waktu eksekusi algoritma sorting. Function `timeSort()` menggunakan `chrono::high_resolution_clock` untuk mendapatkan waktu yang presisi hingga 10 digit decimal. Setelah proses sorting selesai, program menampilkan data yang telah diurutkan dan waktu eksekusi yang dibutuhkan.

Pada akhir setiap operasi sorting, program akan menampilkan pesan "Press any key to continue..." dan menunggu input dari user menggunakan `getch()`. Setelah user menekan tombol apapun, layar akan dibersihkan menggunakan `system("cls")` dan menu utama akan ditampilkan kembali. Loop ini akan terus berlanjut hingga user memilih opsi 7 untuk keluar dari program.

## Analisis Algoritma Insertion Sort

Algoritma Insertion Sort bekerja dengan cara mengambil elemen satu per satu dari bagian yang belum terurut, kemudian menyisipkannya ke posisi yang tepat di bagian yang sudah terurut. Dalam implementasi ini, algoritma dimulai dari indeks ke-1 ( $i=1$ ) dan menggunakan variabel *key* untuk menyimpan karakter yang akan disisipkan.

Proses sorting dimulai dengan mengambil karakter pada posisi *i* sebagai *key*, kemudian membandingkannya dengan karakter-karakter sebelumnya ( $j = i-1$ ). Jika karakter sebelumnya lebih besar dari *key*, maka karakter tersebut akan digeser ke kanan. Proses pergeseran ini berlanjut hingga ditemukan posisi yang tepat untuk *key* atau hingga mencapai awal array.

Kelebihan dari Insertion Sort adalah algoritma ini sangat efisien untuk data berukuran kecil dan memiliki data sudah hampir terurut. Algoritma ini juga bersifat *stable*, mempertahankan urutan relatif elemen yang memiliki nilai sama. Selain itu, Insertion Sort merupakan *in-place sorting algorithm* yang hanya membutuhkan  $O(1)$  extra memory space.

Kekurangan utama Insertion Sort terletak pada time complexity yang kurang efisien untuk data berukuran besar. Pada worst case (data terurut terbalik), algoritma ini memiliki time complexity  $O(n^2)$  karena setiap elemen harus dibandingkan dengan semua elemen sebelumnya. Average case juga memiliki time complexity  $O(n^2)$ , sedangkan best case (data sudah terurut) memiliki time complexity  $O(n)$ .

## Analisis Algoritma Merge Sort

Algoritma Merge Sort menggunakan pendekatan membagi array menjadi dua bagian secara rekursif hingga mencapai elemen tunggal, kemudian menggabungkan kembali dengan cara yang terurut. Implementasi ini menggunakan dua fungsi utama: `mergeSort()` untuk membagi array dan `merge()` untuk menggabungkan sub-array.

Proses sorting dimulai dengan menentukan titik tengah array menggunakan formula  $\text{mid} = \text{left} + (\text{right} - \text{left}) / 2$ . Kemudian fungsi `mergeSort()` dipanggil secara rekursif untuk bagian kiri dan kanan. Setelah kedua bagian telah terurut, fungsi `merge()` akan menggabungkan kedua sub-array dengan membandingkan elemen-elemen dari kedua bagian dan menyusunnya secara ascending.

Dalam fungsi `merge()`, program menggunakan dynamic memory allocation untuk membuat temporary array `tempL` dan `tempR` yang menyimpan data dari sub-array kiri dan kanan. Proses merging dilakukan dengan membandingkan elemen terdepan dari kedua temporary array, kemudian memilih yang lebih kecil untuk dimasukkan ke array utama. Setelah salah satu array habis, sisa elemen dari array lainnya akan disalin ke array utama.

Kelebihan utama Merge Sort adalah time complexity yang stabil  $O(n \log n)$  pada semua kasus (best, average, dan worst case). Algoritma ini juga bersifat stable dan memiliki performa yang predictable. Merge Sort sangat cocok untuk data berukuran besar dan dapat diparalelkan dengan mudah karena sifat divide and conquer-nya.

Kekurangan Merge Sort terletak pada space complexity yang tinggi, yaitu  $O(n)$  karena membutuhkan additional memory untuk temporary arrays. Algoritma ini juga tidak bersifat in-place, sehingga membutuhkan memory tambahan yang signifikan. Untuk data berukuran kecil, overhead dari recursive calls dan memory allocation dapat membuat performanya lebih lambat dibanding algoritma sederhana seperti Insertion Sort.

## Analisis Algoritma Shell Sort

Algoritma Shell Sort merupakan pengembangan dari Insertion Sort yang menggunakan konsep gap sequence untuk mengurangi jumlah perbandingan. Implementasi ini menggunakan gap sequence yang dimulai dari  $n/2$  dan terus dibagi dua hingga mencapai 1. Pada setiap gap, algoritma melakukan insertion sort pada elemen-elemen yang berjarak gap.

Proses sorting dimulai dengan  $\text{gap} = n/2$ , kemudian melakukan insertion sort pada sub-array yang terdiri dari elemen-elemen dengan jarak gap. Misalnya dengan

gap = 4, elemen pada indeks 0, 4, 8, 12 akan diurutkan terlebih dahulu, kemudian elemen pada indeks 1, 5, 9, 13, dan seterusnya. Setelah semua sub-array dengan gap tertentu selesai diurutkan, gap akan diperkecil menjadi  $gap/2$  dan proses diulangi.

Ketika gap = 1, Shell Sort akan melakukan insertion sort pada seluruh array. Namun pada tahap ini, array sudah hampir terurut karena proses-proses sebelumnya, sehingga insertion sort akan berjalan dengan sangat efisien. Inilah yang membuat Shell Sort lebih cepat dari Insertion Sort biasa.

Kelebihan Shell Sort adalah time complexity yang lebih baik dari  $O(n^2)$  pada kebanyakan kasus praktis. Dengan gap sequence yang tepat, Shell Sort dapat mencapai time complexity  $O(n^{1.5})$  atau bahkan lebih baik. Algoritma ini juga bersifat in-place dan tidak membutuhkan extra memory yang signifikan. Shell Sort juga adaptive, performanya akan lebih baik pada data yang sudah hampir terurut.

Kekurangan Shell Sort terletak pada time complexity yang bergantung pada gap sequence yang digunakan. Pemilihan gap sequence yang tidak optimal dapat membuat performanya mendekati  $O(n^2)$ . Algoritma ini juga tidak stable, sehingga urutan relatif elemen dengan nilai sama tidak terjamin. Analisis time complexity Shell Sort juga lebih kompleks dibanding algoritma lainnya.

### **Analisis Algoritma Bubble Sort**

Algoritma Bubble Sort bekerja dengan cara membandingkan pasangan elemen yang bersebelahan dan menukarnya jika urutannya salah. Proses ini diulang secara berulang hingga tidak ada lagi pertukaran yang diperlukan. Implementasi ini menggunakan optimasi dengan flag swapped untuk mendeteksi apakah masih ada pertukaran pada iterasi tertentu.

Proses sorting dimulai dengan loop luar yang berjalan sebanyak  $n-1$  kali, di mana  $n$  adalah jumlah elemen. Pada setiap iterasi loop luar, loop dalam akan membandingkan elemen-elemen bersebelahan dari awal hingga akhir yang belum terurut. Jika elemen kiri lebih besar dari elemen kanan, kedua elemen akan ditukar dan flag swapped akan di-set menjadi true.

Optimasi yang digunakan adalah early termination, yaitu jika pada suatu iterasi tidak ada pertukaran yang terjadi (swapped = false), berarti array sudah terurut dan proses sorting dapat dihentikan. Hal ini membuat Bubble Sort memiliki best case time complexity  $O(n)$  ketika data sudah terurut, meskipun average dan worst case tetap  $O(n^2)$ .

Kelebihan Bubble Sort adalah kesederhanaannya yang membuatnya mudah dipahami dan diimplementasikan. Algoritma ini juga bersifat stable dan in-place, serta dapat mendeteksi apakah array sudah terurut dengan menggunakan flag optimasi. Bubble Sort juga memiliki adaptive property pada implementasi yang dioptimasi.

Kekurangan utama Bubble Sort adalah time complexity  $O(n^2)$  yang membuatnya tidak efisien untuk data berukuran besar. Algoritma ini juga melakukan banyak pertukaran yang tidak perlu, sehingga performanya lebih lambat dibanding algoritma  $O(n^2)$  lainnya seperti Selection Sort. Bubble Sort juga tidak cocok untuk aplikasi yang membutuhkan performa tinggi.

### **Analisis Algoritma Quick Sort**

Algoritma Quick Sort menggunakan strategi divide and conquer dengan memilih satu elemen sebagai pivot, kemudian mempartisi array sehingga elemen yang lebih kecil dari pivot berada di sebelah kiri dan elemen yang lebih besar berada di sebelah kanan. Implementasi ini menggunakan elemen terakhir sebagai pivot dan menggunakan Lomuto partition scheme.

Proses sorting dimulai dengan memanggil fungsi `partition()` yang akan menentukan posisi akhir pivot setelah partitioning. Dalam fungsi `partition`, variabel `i` digunakan untuk melacak batas antara elemen yang lebih kecil dan lebih besar dari pivot. Setiap kali ditemukan elemen yang lebih kecil atau sama dengan pivot, elemen tersebut akan ditukar ke bagian kiri array dan `i` akan diincrement.

Setelah partitioning selesai, pivot akan ditempatkan pada posisi yang tepat dengan menukar elemen pada posisi `i+1` dengan pivot. Kemudian fungsi `quickSort()` akan dipanggil secara rekursif untuk sub-array kiri (low to pivot-1) dan sub-array kanan (pivot+1 to high). Proses rekursi akan berhenti ketika sub-array hanya memiliki satu elemen atau kosong.

Kelebihan utama Quick Sort adalah average case time complexity yang sangat baik yaitu  $O(n \log n)$ , dan biasanya lebih cepat dari algoritma  $O(n \log n)$  lainnya dalam praktik. Algoritma ini juga bersifat in-place dan memiliki space complexity  $O(\log n)$  karena recursive calls. Quick Sort juga memiliki good cache locality yang membuatnya efisien pada sistem modern.

Kekurangan Quick Sort terletak pada worst case time complexity  $O(n^2)$  yang terjadi ketika pivot yang dipilih selalu merupakan elemen terkecil atau terbesar. Algoritma ini juga tidak stable dan performanya sangat bergantung pada pemilihan pivot. Pada implementasi rekursif, Quick Sort dapat menyebabkan stack overflow

untuk data berukuran sangat besar jika tidak dioptimasi dengan tail recursion atau iterative approach.

### **Analisis Algoritma Selection Sort**

Algoritma Selection Sort bekerja dengan cara mencari elemen terkecil dalam array, kemudian menukarnya dengan elemen pertama. Proses ini diulang untuk posisi kedua, ketiga, dan seterusnya hingga seluruh array terurut. Implementasi ini menggunakan variabel `minIndex` untuk melacak posisi elemen terkecil pada setiap iterasi.

Proses sorting dimulai dengan loop luar yang berjalan dari indeks 0 hingga  $n-2$ . Pada setiap iterasi, program akan mencari elemen terkecil dalam sub-array yang dimulai dari posisi  $i$  hingga akhir array. Pencarian dilakukan dengan loop dalam yang membandingkan setiap elemen dengan elemen pada `minIndex` dan mengupdate `minIndex` jika ditemukan elemen yang lebih kecil.

Setelah menemukan elemen terkecil, program akan menukarnya dengan elemen pada posisi  $i$  menggunakan fungsi `swap()`. Proses ini akan berlanjut hingga seluruh array terurut. Karakteristik unik dari Selection Sort adalah jumlah pertukaran yang minimal, yaitu maksimal  $n-1$  kali pertukaran untuk  $n$  elemen.

Kelebihan Selection Sort adalah jumlah pertukaran yang minimal, membuatnya cocok untuk aplikasi di mana operasi swap memiliki cost yang tinggi. Algoritma ini juga bersifat in-place dan memiliki implementasi yang sederhana. Selection Sort juga memiliki performa yang konsisten karena selalu melakukan  $n(n-1)/2$  perbandingan terlepas dari kondisi awal data.

Kekurangan utama Selection Sort adalah time complexity  $O(n^2)$  pada semua kasus, membuatnya tidak efisien untuk data berukuran besar. Algoritma ini juga tidak stable dan tidak adaptive, artinya performanya tidak akan membaik meskipun data sudah hampir terurut. Selection Sort juga melakukan banyak perbandingan yang tidak perlu pada kasus-kasus tertentu.

## MODUL 6 : SEARCHING

### SOAL 1

Ketikkan source code berikut pada program IDE bahasa pemrograman C++  
(Gabungkan 2 code berikut menjadi 1 file (Menu):

- Sequential Searching

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <time.h>
4
5  using namespace std;
6
7  int random(int bil)
8  {
9      int jumlah = rand() % bil;
10     return jumlah;
11 }
12
13 void randomize()
14 {
15     srand(time(NULL));
16 }
17
18 void clrscr()
19 {
20     system("cls");
21 }
22
23 int main()
24 {
25     clrscr();
26     int data[100];
27     int cari = 20;
28     int counter = 0;
29     int flag = 0;
30     int save;
31     randomize();
32     printf("generating 100 number . . .\n");
33     for (int i = 0; i < 100; i++)
34     {
35         data[i] = random(100) + 1;
36         printf("%d ", data[i]);
37     }
38     printf("\ndone.\n");
39
40     for (int i = 0; i < 100; i++)
41     {
42         if (data[i] == cari)
43         {
44             counter++;
45             flag = 1;
46             save = i;
47         }
48     },
49
50     if (flag == 1)
51     {
52         printf("Data ada, sebanyak %d!\n", counter);
53         printf("pada indeks ke-%d", save);
54     }
55     else
56     {
57         printf("Data tidak ada!\n");
58     }
59 }
60
```

Gambar 33. Modul 6 Sequential Searching Soal 1



- Binary Searching

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n, kiri, kanan, tengah, temp, key;
7      bool ketemu = false;
8
9      cout << "Masukan jumlah data? ";
10     cin >> n;
11     int angka[n];
12     for (int i = 0; i < n; i++)
13     {
14         cout << "Angka ke - [" << i << "] : ";
15         cin >> angka[i];
16     }
17
18     for (int i = 0; i < n; i++)
19     {
20         for (int j = 0; j < n - 1; j++)
21         {
22             if (angka[j] > angka[j + 1])
23             {
24                 temp = angka[j];
25                 angka[j] = angka[j + 1];
26                 angka[j + 1] = temp;
27             }
28         }
29     }
30     cout << "-----\n";
31     cout << "Data yang telah diurutkan adalah:\n";
32     for (int i = 0; i < n; i++)
33     {
34         cout << angka[i] << " ";
35     }
36     cout << "\n-----\n";
37     cout << "Masukan angka yang dicari: ";
38     cin >> key;
39
40     kiri = 0;
41     kanan = n - 1;
42     while (kiri <= kanan)
43     {
44         tengah = (kiri + kanan) / 2;
45         if (key == angka[tengah])
46         {
47             ketemu = true;
48             break;
49         }
50         else if (key < angka[tengah])
51         {
52             kanan = tengah - 1;
53         }
54         else
55         {
56             kiri = tengah + 1;
57         }
58     }
59     if (ketemu == true)
60     {
61         cout << "Angka ditemukan! ";
62     }
63     else
64     {
65         cout << "Angka tidak ditemukan!";
66     }
67     return 0;
68 }

```

Gambar 34. Modul 6 Binary Searching Soal 1

## B. Source Code

Tabel 6. Modul 6 Source Code Soal 1

1	#include <iostream>
2	#include <conio.h>
3	#include <random>
4	#include <vector>
5	#include <algorithm>
6	#include <iomanip>
7	using namespace std;
8	
9	// Sequential Search
10	void sequentialSearch(const vector<int>& nums, int target) {
11	vector<int> foundIndices;
12	
13	cout << "\n100 angka acak telah digenerate.\nNums:\n";
14	for (int i = 0; i < nums.size(); i++) {
15	cout << "[" << i << "]: " << nums[i] << " ";
16	}
17	cout << "\n\nMasukkan angka yang ingin dicari: ";
18	cin >> target;
19	
20	for (int i = 0; i < nums.size(); i++) {
21	if (nums[i] == target) {
22	foundIndices.push_back(i);
23	}
24	}
25	
26	if (!foundIndices.empty()) {
27	cout << "\nTarget " << target << " ditemukan sebanyak
28	" << foundIndices.size() << " kali di indeks: ";
29	for (int i : foundIndices) {
30	cout << i << " ";
31	}
32	cout << "\n";
	} else {

33	cout << "Target " << target << " tidak ditemukan dalam data.\n";
34	}
35	}
36	
37	// Binary Search
38	void binarySearch(const vector<int>& numsUnsorted, int target)
39	{
40	vector<int> nums = numsUnsorted;
41	sort(nums.begin(), nums.end());
42	
43	cout << "\n" << nums.size() << " angka telah digenerate dan diurutkan.\nNums:\n";
44	for (int i = 0; i < nums.size(); i++) {
45	cout << "[" << i << "]: " << nums[i] << " ";
46	}
47	
48	cout << "\n\nMasukkan angka yang ingin dicari: ";
49	cin >> target;
50	
51	int kiri = 0, kanan = nums.size() - 1;
52	int posisi = -1;
53	
54	while (kiri <= kanan) {
55	int tengah = (kiri + kanan) / 2;
56	if (nums[tengah] == target) {
57	posisi = tengah;
58	break;
59	} else if (target < nums[tengah]) {
60	kanan = tengah - 1;
61	} else {
62	kiri = tengah + 1;
63	}
64	}
65	
66	if (posisi != -1)
67	cout << "\nTarget " << target << " ditemukan di index = " << posisi << "\n";

```

68         else
69             cout << "\nTarget tidak ditemukan dalam data.\n";
70     }
71
72     void clearScreen() {
73         system("cls");
74     }
75
76     // Penjelasan
77     void explain() {
78         cout << "\n=== PERBEDAAN SEQUENTIAL DAN BINARY SEARCHING
79         ===\n";
80         cout << "Sequential Search:\n";
81         cout << "    - Tidak perlu data terurut.\n";
82         cout << "    - Mencari satu-satu dari awal hingga akhir.\n";
83         cout << "    - Waktu pencarian: O(n).\n\n";
84         cout << "Binary Search:\n";
85         cout << "    - Hanya bisa dilakukan pada data yang sudah
86         terurut.\n";
87         cout << "    - Membagi dua bagian data hingga ketemu.\n";
88         cout << "    - Waktu pencarian: O(log n).\n";
89     }
90
91     int main() {
92         int opt, target;
93
94         do {
95             cout << "\nPilih menu\n";
96             cout << "1. Sequential Searching\n";
97             cout << "2. Binary Searching\n";
98             cout << "3. Jelaskan Perbedaan Sequential Searching
99             dan Binary Searching!\n";
100            cout << "4. Exit\n";
101
102            cout << "Pilih: ";
103            cin >> opt;
104
105            switch (opt) {
106                case 1: {

```

```

103         vector<int> nums(100);
104         mt19937_64 rng(random_device{}());
105         uniform_int_distribution<int> dist(1, 50);
106
107         for (auto& val : nums) {
108             val = dist(rng);
109         }
110
111         sequentialSearch(nums, target);
112         break;
113     }
114
115     case 2: {
116         int size;
117         cout << "Masukkan ukuran vector: ";
118         cin >> size;
119
120         vector<int> nums(size);
121         mt19937_64 rng(random_device{}());
122         uniform_int_distribution<int> dist(1, 100);
123
124         for (auto& val : nums) {
125             val = dist(rng);
126         }
127
128         binarySearch(nums, target);
129         break;
130     }
131
132     case 3:
133         explain();
134         break;
135
136     case 4:
137         cout << "\nTERIMA KASIH\n";
138         cout << "Program was made by Daniel
Noprianto (2410817110010)\n";

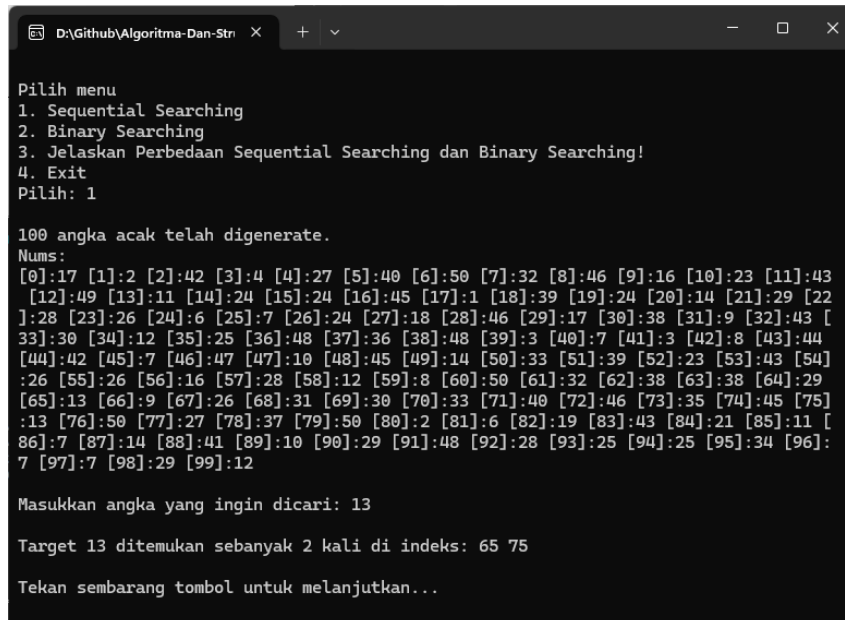
```

```

139         break;
140
141         default:
142             cout << "Opsi tidak valid. Coba lagi.\n";
143     }
144
145     if (opt != 4) {
146         cout << "\nTekan sembarang tombol untuk
melanjutkan...";
147         getch();
148         clearScreen();
149     }
150
151     } while (opt != 4);
152
153     return 0;
154 }

```

### C. Output Program



```

D:\Github\Algoritma-Dan-Str x + v
Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 1

100 angka acak telah digenerate.
Nums:
[0]:17 [1]:2 [2]:42 [3]:4 [4]:27 [5]:40 [6]:50 [7]:32 [8]:46 [9]:16 [10]:23 [11]:43
[12]:49 [13]:11 [14]:24 [15]:24 [16]:45 [17]:1 [18]:39 [19]:24 [20]:14 [21]:29 [22]
]:28 [23]:26 [24]:6 [25]:7 [26]:24 [27]:18 [28]:46 [29]:17 [30]:38 [31]:9 [32]:43 [
33]:30 [34]:12 [35]:25 [36]:48 [37]:36 [38]:48 [39]:3 [40]:7 [41]:3 [42]:8 [43]:44
[44]:42 [45]:7 [46]:47 [47]:10 [48]:45 [49]:14 [50]:33 [51]:39 [52]:23 [53]:43 [54]
:26 [55]:26 [56]:16 [57]:28 [58]:12 [59]:8 [60]:50 [61]:32 [62]:38 [63]:38 [64]:29
[65]:13 [66]:9 [67]:26 [68]:31 [69]:30 [70]:33 [71]:40 [72]:46 [73]:35 [74]:45 [75]
:13 [76]:50 [77]:27 [78]:37 [79]:50 [80]:2 [81]:6 [82]:19 [83]:43 [84]:21 [85]:11 [
86]:7 [87]:14 [88]:41 [89]:10 [90]:29 [91]:48 [92]:28 [93]:25 [94]:25 [95]:34 [96]:
7 [97]:7 [98]:29 [99]:12

Masukkan angka yang ingin dicari: 13

Target 13 ditemukan sebanyak 2 kali di indeks: 65 75

Tekan sembarang tombol untuk melanjutkan...

```

Gambar 35. Modul 6 Output Sequential Searching Saol 1

```
D:\Github\Algoritma-Dan-Stri x + v
Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 2
Masukkan ukuran vector: 6

6 angka telah digenerate dan diurutkan.
Nums:
[0]:4 [1]:7 [2]:34 [3]:54 [4]:80 [5]:83

Masukkan angka yang ingin dicari: 54

Target 54 ditemukan di index = 3

Tekan sembarang tombol untuk melanjutkan...
```

Gambar 36. Modul 6 Output Binary Searching Saol 1

```
D:\Github\Algoritma-Dan-Stri x + v
Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 3

=== PERBEDAAN SEQUENTIAL DAN BINARY SEARCHING ===
Sequential Search:
- Tidak perlu data terurut.
- Mencari satu-satu dari awal hingga akhir.
- Waktu pencarian:  $O(n)$ .

Binary Search:
- Hanya bisa dilakukan pada data yang sudah terurut.
- Membagi dua bagian data hingga ketemu.
- Waktu pencarian:  $O(\log n)$ .

Tekan sembarang tombol untuk melanjutkan...
```

Gambar 37. Modul 6 Output Perbedaan Sequential dan Binary Searching Saol 1

## D. Pembahasan

### Alur Program Sorting

Ketika program dijalankan, sistem akan menampilkan menu utama dengan 4 pilihan yang tersedia menggunakan do-while loop yang terus berjalan hingga user memilih opsi 4 (Exit). Setiap kali user memilih opsi 1-3, program akan melakukan proses sesuai dengan pilihan yang dipilih.

Untuk opsi 1 (Sequential Searching), program akan menggenerate 100 angka acak dengan rentang 1-50 menggunakan random number generator *mt19937\_64* dan *uniform\_int\_distribution*. Data yang dihasilkan disimpan ke vector dan ditampilkan beserta indeksinya. User kemudian diminta untuk memasukkan angka yang ingin dicari, dan program akan mencari semua angka tersebut dalam data.

Untuk opsi 2 (Binary Searching), program akan meminta user untuk memasukkan ukuran vector terlebih dahulu, kemudian menggenerate angka acak sesuai ukuran yang ditentukan dengan rentang 1-100. Data yang dihasilkan akan diurutkan terlebih dahulu menggunakan fungsi *sort()* dari *STL*, kemudian ditampilkan. Setelah itu, user diminta memasukkan angka yang ingin dicari, dan program akan melakukan pencarian menggunakan algoritma binary search.

Untuk opsi 3, program akan menampilkan penjelasan mengenai perbedaan antara Sequential Search dan Binary Search, termasuk karakteristik dan kompleksitas waktu masing-masing algoritma.

Pada akhir setiap operasi (kecuali exit), program akan menampilkan pesan "*Tekan sembarang tombol untuk melanjutkan...*" dan menunggu input menggunakan *getch()*. Setelah user menekan tombol apapun, layar akan dibersihkan menggunakan *system("cls")* dan menu utama akan ditampilkan kembali. Loop ini akan terus berlanjut hingga user memilih opsi 4 untuk keluar dari program.



## Analisis Algoritma Sequential Search

Algoritma Sequential Search (Linear Search) bekerja dengan cara memeriksa setiap elemen dalam array secara berurutan dari awal hingga akhir untuk mencari target yang diinginkan. Implementasi ini menggunakan vector `foundIndices` untuk menyimpan semua indeks di mana target ditemukan, sehingga dapat menangani kasus di mana target muncul lebih dari sekali.

Proses searching dimulai dengan loop `for` yang iterasi dari indeks 0 hingga ukuran vector dikurangi satu. Pada setiap iterasi, program membandingkan elemen pada posisi `i` dengan target yang dicari. Jika elemen tersebut sama dengan target, indeks `i` akan disimpan dalam vector `foundIndices`. Proses ini berlanjut hingga seluruh elemen dalam array telah diperiksa.

Setelah proses pencarian selesai, program akan memeriksa apakah vector `foundIndices` kosong atau tidak. Jika tidak kosong, program akan menampilkan informasi bahwa target ditemukan beserta jumlah kemunculan dan semua indeks di mana target tersebut ditemukan. Jika kosong, program akan menampilkan pesan bahwa target tidak ditemukan dalam data.

Kelebihan dari Sequential Search adalah algoritma ini dapat bekerja pada data yang tidak terurut dan dapat menemukan semua kemunculan target dalam satu kali pencarian. Algoritma ini juga mudah diimplementasikan dan dipahami, serta memiliki space complexity yang rendah yaitu  $O(1)$  jika hanya mencari satu kemunculan. Sequential Search juga tidak memerlukan preprocessing data seperti pengurutan.

Kekurangan utama Sequential Search terletak pada time complexity yang kurang efisien untuk data berukuran besar. Pada worst case (target berada di akhir array atau tidak ada), algoritma ini memiliki time complexity  $O(n)$  karena harus memeriksa semua elemen. Average case juga memiliki time complexity  $O(n/2)$  yang masih tergolong  $O(n)$ , sedangkan best case (target berada di awal array) memiliki time complexity  $O(1)$ .

## Analisis Algoritma Binary Search

Algoritma Binary Search menggunakan strategi divide and conquer dengan cara membagi array menjadi dua bagian secara berulang hingga target ditemukan atau tidak ada lagi elemen yang dapat dibagi. Algoritma ini hanya dapat bekerja pada data yang sudah terurut, sehingga implementasi ini menggunakan fungsi `sort()` terlebih dahulu untuk mengurutkan data.

Proses searching dimulai dengan menentukan batas kiri ( $\text{kiri} = 0$ ) dan batas kanan ( $\text{kanan} = \text{ukuran array} - 1$ ). Kemudian program menghitung posisi tengah menggunakan formula  $\text{tengah} = (\text{kiri} + \text{kanan}) / 2$ . Elemen pada posisi tengah akan dibandingkan dengan target yang dicari.

Jika elemen tengah sama dengan target, maka target telah ditemukan dan posisinya disimpan. Jika target lebih kecil dari elemen tengah, pencarian akan dilanjutkan pada bagian kiri array dengan mengubah  $\text{kanan} = \text{tengah} - 1$ . Jika target lebih besar dari elemen tengah, pencarian akan dilanjutkan pada bagian kanan array dengan mengubah  $\text{kiri} = \text{tengah} + 1$ . Proses ini berlanjut hingga target ditemukan atau  $\text{kiri} > \text{kanan}$  (target tidak ada).

Implementasi ini menggunakan variabel posisi untuk menyimpan indeks di mana target ditemukan. Jika posisi masih bernilai -1 setelah loop selesai, berarti target tidak ditemukan dalam data. Program kemudian akan menampilkan hasil pencarian kepada user.

Kelebihan utama Binary Search adalah time complexity yang sangat efisien yaitu  $O(\log n)$  pada semua kasus (best, average, dan worst case). Algoritma ini sangat cocok untuk data berukuran besar dan memiliki performa yang predictable. Binary Search juga memiliki space complexity  $O(1)$  pada implementasi iterative dan memiliki efisiensi yang tinggi dalam praktik.

Kekurangan Binary Search terletak pada requirement bahwa data harus sudah terurut terlebih dahulu. Jika data belum terurut, diperlukan proses sorting yang memiliki time complexity minimal  $O(n \log n)$ , sehingga untuk pencarian sekali saja mungkin tidak efisien. Algoritma ini juga hanya dapat menemukan satu kemunculan target (biasanya yang pertama ditemukan) dan lebih kompleks untuk diimplementasikan dibanding Sequential Search.

### **Perbandingan Sequential Search dan Binary Search**

Dari segi kompleksitas waktu, Binary Search memiliki keunggulan yang signifikan dengan  $O(\log n)$  dibandingkan Sequential Search yang memiliki  $O(n)$ . Namun, Binary Search memerlukan data yang sudah terurut, sedangkan Sequential Search dapat bekerja pada data apa adanya.

Dari segi penggunaan praktis, Sequential Search lebih cocok untuk dataset kecil atau ketika data sering berubah dan tidak selalu terurut. Binary Search lebih cocok untuk dataset besar yang sudah terurut atau ketika pencarian dilakukan berulang kali pada data yang sama.

Sequential Search juga memiliki keunggulan dalam menemukan semua kemunculan target dalam satu kali pencarian, sedangkan Binary Search biasanya hanya menemukan satu kemunculan. Untuk aplikasi yang memerlukan pencarian multiple occurrences, Sequential Search atau modifikasi Binary Search mungkin lebih sesuai.

## MODUL 7 : TREE

### SOAL 1

Cobalah program berikut, perbaiki *output*, lengkapi fungsi *inOrder* dan *postOrder* pada *coding*, *running*, simpan program !

```
1  #include <stdio.h>
2  #include <conio.h>
3  #include <stdlib.h>
4  #include <iostream>
5
6  using namespace std;
7  struct Node
8  {
9      int data;
10     Node *kiri;
11     Node *kanan;
12 };
13
14 void tambah(Node **root, int databaru)
15 {
16     if (*root == NULL)
17     {
18         Node *baru;
19         baru = new Node;
20         baru->data = databaru;
21         baru->kiri = NULL;
22         baru->kanan = NULL;
23         (*root) = baru;
24         (*root)->kiri = NULL;
25         (*root)->kanan = NULL;
26         cout << "Data bertambah";
27     }
28     else if (databaru < (*root)->data)
29         tambah(&(*root)->kiri, databaru);
30     else if (databaru > (*root)->data)
31         tambah(&(*root)->kanan, databaru);
32     else if (databaru == (*root)->data)
33         cout << "Data sudah ada";
34 }
35
36 void preOrder(Node *root)
37 {
38     if (root != NULL)
39     {
40         cout << root->data;
41         preOrder(root->kiri);
42         preOrder(root->kanan);
43     }
44 }
45
46 void inOrder(Node *root)
47 {
48     if (root != NULL)
```

```

49
50
51
52
53
54 }
55
56 void postOrder(Node *root)
57 {
58
59
60
61
62
63
64 }
65
66 int main()
67 {
68     int pil, data;
69     Node *pohon;
70     pohon = NULL;
71     do
72     {
73         system("cls");
74         cout << "1. Tambah\n";
75         cout << "2. PreOrder\n";
76         cout << "3. inOrder\n";
77         cout << "4. PostOrder\n";
78         cout << "5. Exit\n";
79         cout << "\nPilihan : ";
80         cin >> pil;
81         switch (pil)
82         {
83             case 1:
84                 cout << "\n INPUT : ";
85                 cout << "\n -----";
86                 cout << "\n Data baru : ";
87                 cin >> data;
88                 tambah(&pohon, data);
89                 break;
90             case 2:
91                 cout << "PreOrder";
92                 cout << "\n-----\n";
93                 if (pohon != NULL)
94                 {
95                     preOrder(pohon);
96

```

```

97         else
98             cout << "Masih Kosong";
99         break;
100     case 3:
101         cout << "InOrder";
102         cout << "\n-----\n";
103         if (pohon != NULL)
104         {
105             inOrder(pohon);
106         }
107         else
108             cout << "Masih Kosong";
109         break;
110     case 4:
111         cout << "PostOrder";
112         cout << "\n-----\n";
113         if (pohon != NULL)
114         {
115             postOrder(pohon);
116         }
117         else
118             cout << "Masih Kosong";
119         break;
120     case 5:
121         return 0;
122     }
123     _getch();
124 } while (pil != 5);
125 return EXIT_FAILURE;
126 }

```

Gambar 38. Modul 6 Soal 1

## B. Source Code

Tabel 7. Modul 7 Source Code Soal 1

1	#include <iostream>
2	#include <conio.h>
3	#include <stdlib.h>
4	
5	using namespace std;
6	
7	struct Node {
8	int data;
9	Node *left;
10	Node *right;

```

11  };
12
13  void insert(Node **root, int newData){
14      if (*root == nullptr){
15          Node *newNode;
16          newNode = new Node;
17
18          newNode -> data = newData;
19          newNode -> left = nullptr;
20          newNode -> right = nullptr;
21
22          *root = newNode;
23
24          cout << "Data has been added\n";
25      }
26      else if (newData < (*root) -> data){
27          insert(&((*root)->left), newData);
28      }
29      else if (newData > (*root) -> data){
30          insert(&((*root)->right), newData);
31      }
32      else if (newData == (*root) -> data){
33          cout << "Data is already exist\n";
34      }
35  }
36
37  void preOrder(Node *root){
38      if (root != NULL){
39          cout << root->data << " ";
40          preOrder(root->left);
41          preOrder(root->right);
42      }
43  }
44
45  void inOrder(Node *root){
46      if (root != NULL){

```

```

47     inOrder(root->left);
48     cout << root->data << " ";
49     inOrder(root->right);
50 }
51 }
52
53 void postOrder(Node *root){
54     if (root != NULL){
55         postOrder(root->left);
56         postOrder(root->right);
57         cout << root->data << " ";
58     }
59 }
60
61 void destroyTree(Node *root) {
62     if (root != nullptr) {
63         destroyTree(root->left);
64         destroyTree(root->right);
65         delete root;
66     }
67 }
68
69 int main(){
70     int opt, val;
71     Node *tree;
72     tree = NULL;
73
74     do {
75         system("cls");
76
77         cout << "1. Insert\n";
78         cout << "2. PreOrder\n";
79         cout << "3. InOrder\n";
80         cout << "4. PostOrder\n";
81         cout << "5. Exit\n";
82

```



```

83     cout << "\nOption: "; cin >> opt;
84
85     switch (opt) {
86
87         case 1:
88             cout << "\n Input:";
89             cout << "\n -----";
90             cout << "\n New data: ";
91             cin >> val;
92             insert(&tree, val);
93             break;
94
95         case 2:
96             cout << "PreOrder Traversal\n";
97             cout << "=====\n";
98             if (tree == NULL) {
99                 cout << "Tree is empty!\n";
100            }
101            else {
102                preOrder(tree);
103            }
104            break;
105
106         case 3:
107             cout << "InOrder Traversal\n";
108             cout << "=====\n";
109             if (tree == NULL) {
110                 cout << "Tree is empty!\n";
111            }
112            else {
113                inOrder(tree);
114            }
115            break;
116
117         case 4:
118             cout << "PostOrder Traversal\n";

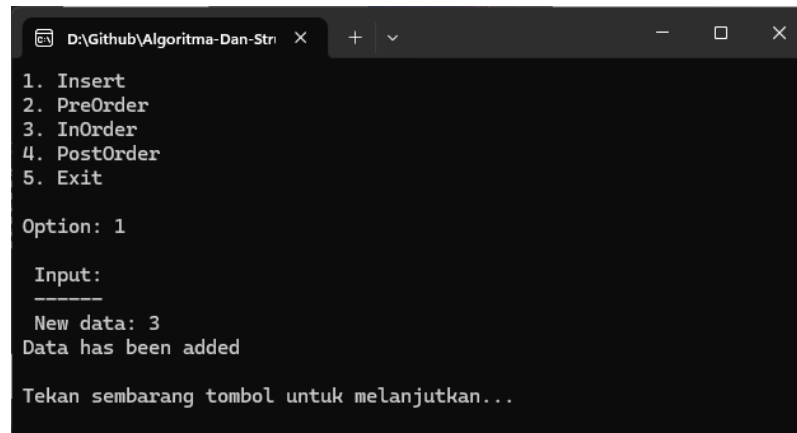
```

```

119         cout << "=====\n";
120         if (tree == NULL) {
121             cout << "Tree is empty!\n";
122         }
123         else {
124             postOrder(tree);
125         }
126         break;
127
128         case 5:
129             cout << "\nExiting program and cleaning up
memory...\n";
130             destroyTree(tree);
131             tree = nullptr;
132             cout << "Memory cleaned up. Goodbye!\n";
133             break;
134
135             default:
136                 cout << "Option is not valid! Please re-enter
your option";
137                 break;
138         }
139
140         if (opt != 5) {
141             cout << "\nTekan sembarang tombol untuk
melanjutkan...";
142             getch();
143         }
144
145     } while(opt != 5);
146     return 0;
147 }

```

### C. Output Program



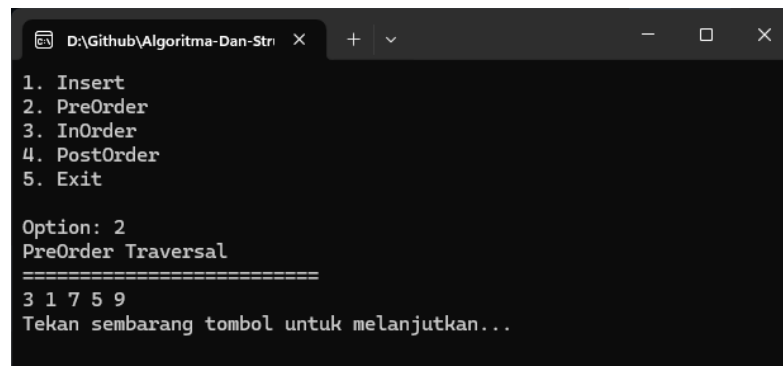
```
D:\Github\Algoritma-Dan-Stri X + - □ X
1. Insert
2. PreOrder
3. InOrder
4. PostOrder
5. Exit

Option: 1

Input:
-----
New data: 3
Data has been added

Tekan sembarang tombol untuk melanjutkan...
```

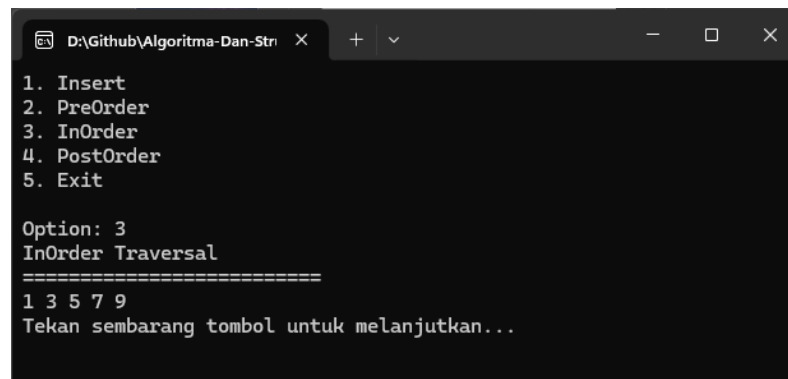
Gambar 39. Modul 7 Output Insert Saol 1



```
D:\Github\Algoritma-Dan-Stri X + - □ X
1. Insert
2. PreOrder
3. InOrder
4. PostOrder
5. Exit

Option: 2
PreOrder Traversal
=====
3 1 7 5 9
Tekan sembarang tombol untuk melanjutkan...
```

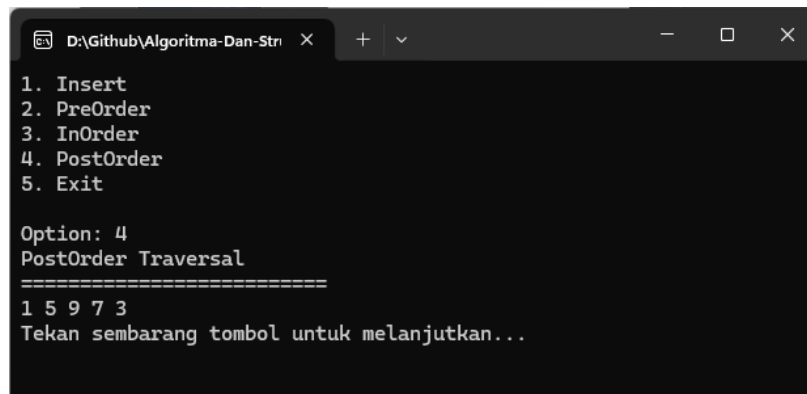
Gambar 40. Modul 7 Output PreOrder Saol 1



```
D:\Github\Algoritma-Dan-Stri X + - □ X
1. Insert
2. PreOrder
3. InOrder
4. PostOrder
5. Exit

Option: 3
InOrder Traversal
=====
1 3 5 7 9
Tekan sembarang tombol untuk melanjutkan...
```

Gambar 41. Modul 7 Output InOrder Saol 1



```
D:\Github\Algoritma-Dan-Stri x + v - □ X
1. Insert
2. PreOrder
3. InOrder
4. PostOrder
5. Exit

Option: 4
PostOrder Traversal
=====
1 5 9 7 3
Tekan sembarang tombol untuk melanjutkan...
```

Gambar 42. Modul 7 Output PostOrder Saol 1

## D. Pembahasan

### Alur Program Binary Search Tree

Ketika program dijalankan, sistem akan menampilkan menu utama dengan 5 pilihan yang tersedia menggunakan do-while loop yang terus berjalan hingga user memilih opsi 5 (Exit). Setiap kali user memilih opsi 1-4, program akan melakukan proses sesuai dengan pilihan yang dipilih, kemudian layar akan dibersihkan menggunakan `system("cls")` dan menu utama akan ditampilkan kembali.

Untuk opsi 1 (Insert), program akan meminta user untuk memasukkan data baru yang ingin ditambahkan ke dalam tree. Program menggunakan fungsi `insert()` yang bekerja secara rekursif untuk menemukan posisi yang tepat sesuai dengan aturan BST (Binary Search Tree). Data yang lebih kecil akan ditempatkan di subtree kiri, sedangkan data yang lebih besar akan ditempatkan di subtree kanan. Jika data yang dimasukkan sudah ada dalam tree, program akan menampilkan pesan "Data is already exist".

Untuk opsi 2, 3, dan 4 (PreOrder, InOrder, PostOrder), program akan melakukan traversal atau penelusuran tree menggunakan tiga metode yang berbeda. Sebelum melakukan traversal, program akan memeriksa apakah tree kosong. Jika kosong, program akan menampilkan pesan "Tree is empty!". Jika tidak kosong, program akan menampilkan hasil traversal sesuai dengan metode yang dipilih.

Untuk opsi 5, program akan melakukan cleanup memory dengan memanggil fungsi `destroyTree()` untuk menghapus semua node yang telah dialokasikan secara dinamis, kemudian mengatur pointer tree menjadi `nullptr` sebelum mengakhiri program. Hal ini dilakukan untuk mencegah memory leak.

Pada akhir setiap operasi (kecuali `exit`), program akan menampilkan pesan "Tekan sembarang tombol untuk melanjutkan..." dan menunggu input menggunakan `getch()`. Setelah user menekan tombol apapun, layar akan dibersihkan dan menu utama akan ditampilkan kembali. Loop ini akan terus berlanjut hingga user memilih opsi 5 untuk keluar dari program.

### **Analisis Struktur Data Binary Search Tree**

Binary Search Tree (BST) adalah struktur data tree berbasis node di mana setiap node memiliki maksimal dua child (anak), yaitu left child dan right child. Struktur Node dalam implementasi ini terdiri dari tiga komponen: data bertipe integer yang menyimpan nilai, dan dua pointer (left dan right) yang menunjuk ke child nodes.

Aturan fundamental BST adalah bahwa untuk setiap node, semua nilai di subtree kiri harus lebih kecil dari nilai node tersebut, dan semua nilai di subtree kanan harus lebih besar dari nilai node tersebut. Aturan ini memungkinkan operasi pencarian, penyisipan, dan penghapusan dapat dilakukan dengan efisien.

Program ini menggunakan implementasi pointer-to-pointer (double pointer) pada fungsi `insert()` untuk memungkinkan modifikasi langsung terhadap pointer root. Hal ini memungkinkan fungsi untuk mengubah nilai pointer yang diteruskan dari fungsi pemanggil, yang sangat penting ketika menambahkan node pertama ke tree yang kosong.

## **Analisis Algoritma Insert**

Algoritma insert() bekerja secara rekursif dengan menggunakan prinsip divide and conquer. Proses dimulai dengan memeriksa apakah root tree kosong (nullptr). Jika kosong, program akan membuat node baru menggunakan dynamic memory allocation dengan operator new, mengisi data node tersebut, dan mengatur kedua pointer child menjadi nullptr.

Jika tree tidak kosong, program akan membandingkan data baru dengan data pada node saat ini. Jika data baru lebih kecil, fungsi akan memanggil dirinya sendiri secara rekursif untuk subtree kiri. Jika data baru lebih besar, fungsi akan memanggil dirinya sendiri secara rekursif untuk subtree kanan. Jika data baru sama dengan data yang sudah ada, program akan menampilkan pesan bahwa data sudah exist dan tidak melakukan penyisipan.

Kelebihan dari algoritma insert ini adalah dapat mempertahankan properti BST secara otomatis dan memiliki implementasi yang elegant menggunakan rekursi. Time complexity untuk operasi insert pada average case adalah  $O(\log n)$ , di mana  $n$  adalah jumlah node dalam tree. Namun pada worst case (ketika tree menjadi skewed atau tidak seimbang), time complexity bisa mencapai  $O(n)$ .

Space complexity untuk operasi insert adalah  $O(h)$  di mana  $h$  adalah height dari tree, karena menggunakan recursive call stack. Pada balanced tree, space complexity adalah  $O(\log n)$ , namun pada skewed tree bisa mencapai  $O(n)$ .

## **Analisis Algoritma Traversal**

Program implementasi ini menyediakan tiga metode traversal yang berbeda: PreOrder, InOrder, dan PostOrder. Ketiga algoritma ini menggunakan pendekatan rekursif untuk mengunjungi setiap node dalam tree dengan urutan yang berbeda.

### **PreOrder Traversal (Root-Left-Right)**

Algoritma PreOrder mengunjungi node dalam urutan: root, kemudian subtree kiri, lalu subtree kanan. Proses dimulai dengan mencetak data node saat ini, kemudian memanggil `preOrder()` secara rekursif untuk left child, dan terakhir memanggil `preOrder()` secara rekursif untuk right child. Traversal ini berguna untuk membuat copy dari tree atau untuk mengevaluasi expression tree.

### **InOrder Traversal (Left-Root-Right)**

Algoritma InOrder mengunjungi node dalam urutan: subtree kiri, root, kemudian subtree kanan. Proses dimulai dengan memanggil `inOrder()` secara rekursif untuk left child, kemudian mencetak data node saat ini, dan terakhir memanggil `inOrder()` secara rekursif untuk right child. Keunggulan utama InOrder traversal pada BST adalah menghasilkan output yang terurut secara ascending, sehingga sangat berguna untuk mendapatkan data dalam urutan yang sorted.

### **PostOrder Traversal (Left-Right-Root)**

Algoritma PostOrder mengunjungi node dalam urutan: subtree kiri, subtree kanan, kemudian root. Proses dimulai dengan memanggil `postOrder()` secara rekursif untuk left child, kemudian memanggil `postOrder()` secara rekursif untuk right child, dan terakhir mencetak data node saat ini. Traversal ini sangat berguna untuk operasi penghapusan tree karena memproses children sebelum parent, sehingga aman untuk menghapus node tanpa kehilangan referensi ke children.

Time complexity untuk semua metode traversal adalah  $O(n)$  karena setiap node harus dikunjungi tepat satu kali. Space complexity adalah  $O(h)$  dimana  $h$  adalah height tree, karena menggunakan recursive call stack yang dalam worst case bisa mencapai height maksimum tree.

## **LINK GITHUB**

<https://github.com/IndraSuryadilaga/Algoritma-Dan-Struktur-Data>