

**LAPORAN AKHIR PRAKTIKUM
ALGORITMA & STRUKTUR DATA**



Oleh:

Rizki Adhitiya Maulana

NIM. 2410817110014

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
2025**

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM ALGORITMA &
STRUKTUR DATA

Laporan Praktikum Algoritma & Struktur Data

Modul 1 : Struct dan Pointer

Modul 2 : Stack dan Queue

Modul 3 : Single Link List

Modul 4 : Double Link List

Modul 5 : Sorting

Modul 6 : Searching

Modul 7 : Tree (Pohon)

Ini disusun sebagai syarat lulus mata kuliah Praktikum Algoritma & Struktur Data.

Laporan Akhir Praktikum ini dikerjakan oleh:

Nama Praktikan : Rizki Adhitiya Maulana

NIM : 2410817110014

Menyetuji,

Asisten Praktikum

Mengetahui,

Dosen Penanggung Jawab Praktikum

Muhammad Fauzan Ahsani

NIM. 2310817310009

Muti'a Maulida, S.Kom., M.TI.

NIP. 198810272019032013

DAFTAR ISI

LEMBAR PENGESAHAN	ii
DAFTAR ISI	iii
DAFTAR GAMBAR	vi
DAFTAR TABEL.....	xi
MODUL 1 : STRUCT DAN POINTER	12
SOAL 1	12
A. Source Code	13
B. Output Program	15
C. Pembahasan	15
SOAL 2	19
A. Source Code	19
B. Output Program	21
C. Pembahasan	21
SOAL 3	24
A. Source Code	24
B. Output Program	25
C. Pembahasan	25
MODUL 2 : STACK DAN QUEUE	28
SOAL 1	28
A. Pembahasan.....	28
SOAL 2	29
A. Source Code	30
B. Output Program	35
C. Pembahasan	38
SOAL 3	41
A. Source Code	42
B. Output Program	47
C. Pembahasan	50
MODUL 3 : SINGLE LINK LIST	53
SOAL 1	53
A. Source Code	55

B. Output Program	72
C. Pembahasan	79
SOAL 2	83
A. Output Program.....	83
B. Pembahasan	83
SOAL 3	84
A. Output Program.....	84
B. Pembahasan	84
SOAL 4	85
A. Output Program.....	85
B. Pembahasan	85
SOAL 5	86
A. Output Program.....	86
B. Pembahasan	86
SOAL 6	87
A. Output Program.....	87
B. Pembahasan	87
SOAL 7	88
A. Output Program.....	88
B. Pembahasan	88
SOAL 8	89
A. Output Program.....	89
B. Pembahasan	89
SOAL 9	90
A. Pembahasan.....	90
SOAL 10	91
A. Pembahasan.....	91
MODUL 4 :DOUBLE LINK LIST	92
SOAL 1	92
A. Source Code	95
B. Output Program	107
C. Pembahasan	115

SOAL 2	120
A. Pembahasan.....	120
SOAL 3	121
A. Pembahasan.....	121
SOAL 4	122
A. Output Program.....	122
B. Pembahasan	124
MODUL 5 : SORTING	125
SOAL 1	125
A. Source Code	126
B. Output Program	136
C. Pembahasan	143
MODUL 6 : SEARCHING	155
SOAL 1	155
A. Source Code	157
B. Output Program	165
C. Pembahasan	167
MODUL 7 : TREE (POHON).....	173
SOAL 1	173
A. Source Code	175
B. Output Program	180
C. Pembahasan	182
TAUTAN GIF	188

DAFTAR GAMBAR

Gambar 1. Soal 1 Modul 1	12
Gambar 2. Screenshot Output Program Soal 1 Modul 1.....	15
Gambar 3. Screenshot Output Program Soal 2 Modul 1.....	21
Gambar 4 Screenshot Output Program Soal 3 Modul 1.....	25
Gambar 5 Soal 2 Modul 2	29
Gambar 6. Tampilan Awal Program Saat Dijalankan	35
Gambar 7. Memasukkan Nilai Ke Dalam Stack	35
Gambar 8. Menampilkan Stack Yang Sudah Dimasukkan	35
Gambar 9. Melakukan Pop Pada Nilai Di Dalam Stack	36
Gambar 10. Tampilan Setelah Nilai Teratas Di Pop	36
Gambar 11. Membersihkan Stack Dengan Pilihan Menu 4	36
Gambar 12. Tampilan Setelah Dibersihkan.....	37
Gambar 13. Tampilan Jika Stack Sudah Penuh.....	37
Gambar 14. Tampilan Saat Memilih Menu 5 Quit.....	37
Gambar 15. Soal 3 Modul 2	41
Gambar 16. Tampilan Awal Program Saat Dijalankan.....	47
Gambar 17. Memasukkan Huruf ke Dalam Queue	47
Gambar 18. Menampilkan Queue Yang Sudah Dimasukkan.....	47
Gambar 19. Melakukan Delete Pada Huruf yang Sudah Dimasukkan	48
Gambar 20. Tampilan Queue Setelah Melakukan Delete	48
Gambar 21. Mereset Queue.....	48
Gambar 22. Tampilan Queue Setelah di Reset.....	49
Gambar 23. Tampilan Jika Queue Penuh	49
Gambar 24. Tampilan Saat Memilih Menu 5 Quit.....	49
Gambar 25 Soal 1 Modul 3	55
Gambar 26. Tampilan Awal Program	72
Gambar 27. Tampilan Tambah Data dari Depan	73
Gambar 28. Tampilan Data Setelah Dilakukan Tambah Data Depan	73
Gambar 29. Tampilan Tambah Data dari Belakang	73
Gambar 30. Tampilan Data Setelah Dilakukan Tambah Data Belakang	74
Gambar 31. Tampilan Cari Data Yang Tidak Ada.....	74

Gambar 32. Tampilan Cari Data Yang Ada	74
Gambar 33. Tampilan Hapus Data Belakang	75
Gambar 34. Tampilan Data Setelah Dilakukan Hapus Data Belakang	75
Gambar 35. Tampilan Hapus Setiap Data Tertentu	75
Gambar 36. Tampilan Data Setelah Dilakukan Hapus Setiap Data Tertentu	76
Gambar 37. Tampilan Hapus Data Depan	76
Gambar 38. Tampilan Data Setelah Dilakukan Hapus Data Depan	76
Gambar 39. Tampilan Sisipkan Node Sebelum Data Tertentu	77
Gambar 40. Tampilan Data Setelah Dilakukan Sisipkan Node Sebelum	77
Gambar 41. Tampilan Sisipkan Node Setelah Data Tertentu	77
Gambar 42. Tampilan Data Setelah Dilakukan Sisipkan Node Setelah	78
Gambar 43. Tampilan Hapus Semua Elemen	78
Gambar 44. Tampilan Data Setelah Dilakukan Hapus Semua Elemen	78
Gambar 45. Tampilan Setelah Selesai Menyelesaikan Program	79
Gambar 46. Tampilan Nilai yang Diinput dari Depan	83
Gambar 47. Tampilan Nilai yang Diinput dari Belakang	84
Gambar 48. Pencarian Nilai 2 pada Linked List	85
Gambar 49. Pencarian Nilai 7 pada Linked List	86
Gambar 50. Menghapus Data dari Belakang Linked List	87
Gambar 51. Tampilan Linked List Setelah Dihapus	87
Gambar 52. Menghapus Setiap Nilai 3 yang Ada di Linked List	88
Gambar 53. Tampilan Linked List Setelah Setiap Nilai 3 Dihapus	88
Gambar 54. Tampilan Isi dari Linked List	89
Gambar 55. Soal 5 Modul 4	95
Gambar 56. Tampilan Menu Double Linked List Non Circular	107
Gambar 57. Masuk Ke Tampilan Menu Head DLLNC	107
Gambar 58. Tambah Data Dari Depan	108
Gambar 59. Tampilan Data Setelah Dilakukan Tambah Depan	108
Gambar 60. Tambah Data Dari Belakang	108
Gambar 61. Tampilan Data Setelah Dilakukan Tambah Belakang	109
Gambar 62. Hapus Data Dari Depan	109
Gambar 63. Tampilan Data Setelah Dilakukan Hapus Depan	109

Gambar 64. Hapus Data Dari Belakang.....	110
Gambar 65. Tampilan Data Setelah Dilakukan Hapus Belakang.....	110
Gambar 66. Reset Data Yang Ada.....	110
Gambar 67. Tampilan Data Setelah Dilakukan Reset Data	111
Gambar 68. Masuk Ke Tampilan Menu Head Dan Tail DLLNC	111
Gambar 69. Tambah Data Dari Depan	111
Gambar 70. Tampilan Data Setelah Dilakukan Tambah Depan.....	112
Gambar 71. Tambah Data Dari Belakang	112
Gambar 72. Tampilan Data Setelah Dilakukan Tambah Belakang	112
Gambar 73. Hapus Data Dari Depan.....	113
Gambar 74. Tampilan Data Setelah Dilakukan Hapus Depan	113
Gambar 75. Hapus Data Dari Belakang.....	113
Gambar 76. Tampilan Data Setelah Dilakukan Hapus Belakang.....	114
Gambar 77. Reset Data Yang Ada.....	114
Gambar 78. Tampilan Data Setelah Dilakukan Reset Data	114
Gambar 79. Tampilan Keluar dari Program	115
Gambar 80. Fungsi Next Di Single Linked List	120
Gambar 81. Fungsi Prev Di Double Linked List	121
Gambar 82. Tampilan Tambah Depan Head	122
Gambar 83. Tampilan Tambah Belakang Head.....	122
Gambar 84. Tampilan Data Head.....	123
Gambar 85. Tampilan Tambah Depan Head Dan Tail	123
Gambar 86. Tampilan Tambah Belakang Head Dan Tail.....	123
Gambar 87. Tampilan Data Head Dan Tail	124
Gambar 88. Soal 1 Modul 5	125
Gambar 89. Tampilan Menu Sorting.....	136
Gambar 90. Pilihan 1 Pada Menu Sorting.....	136
Gambar 91. Masukkan Nama Pada Pilihan 1.....	136
Gambar 92. Tampilan Hasil Dari Insertion Sort	137
Gambar 93. Pilihan 2 Pada Menu Sorting.....	137
Gambar 94. Masukkan Nama Pada Pilihan 2.....	137
Gambar 95. Tampilan Hasil Dari Merge Sort	138

Gambar 96. Pilihan 3 Pada Menu Sorting.....	138
Gambar 97. Masukkan Nama Pada Pilihan 3.....	138
Gambar 98. Tampilan Hasil Dari Shell Sort	139
Gambar 99. Pilihan 4 Pada Menu Sorting.....	139
Gambar 100. Masukkan ID Pada Pilihan 4	139
Gambar 101. Tampilan Hasil Dari Bubble Sort	140
Gambar 102. Pilihan 5 Pada Menu Sorting.....	140
Gambar 103. Masukkan ID Pada Pilihan 5	140
Gambar 104. Tampilan Hasil Dari Quick Sort.....	141
Gambar 105. Pilihan 6 Pada Menu Sorting.....	141
Gambar 106. Masukkan ID Pada Pilihan 6	141
Gambar 107. Tampilan Hasil Dari Selection Sort.....	142
Gambar 108. Pilihan 7 Pada Menu Sorting.....	142
Gambar 109. Tampilan Program Selesai.....	142
Gambar 110. Ilustrasi Insertion Sort	144
Gambar 111. Ilustrasi Merge Sort	146
Gambar 112. Ilustrasi Shell Sort	148
Gambar 113. Ilustrasi Bubblu Sort.....	150
Gambar 114. Ilustrasi Quick Sort.....	152
Gambar 115. Ilustrasi Selection Sort.....	153
Gambar 116. Soal 1 Sequential Searching Modul 6	155
Gambar 117. Soal 1 Binary Searching Modul 6	156
Gambar 118. Soal 1 Tampilan Menu Program Modul 6	156
Gambar 119. Tampilan Menu Searching.....	165
Gambar 120. Tampilan Sequential Searching Apabila Angka Ditemukan	165
Gambar 121. Tampilan Sequential Searching Apabila Angka Tidak Ditemukan	165
Gambar 122. Tampilan Binary Searching Apabila Angka Ditemukan	166
Gambar 123. Tampilan Binary Searching Apabila Angka Tidak Ditemukan	166
Gambar 124. Tampilan Penjelasan Perbedaan Dua Metode Search	166
Gambar 125. Ilustrasi Sequential Search	169
Gambar 126 Ilustrasi Binary Search	170
Gambar 127. Soal 1 Modul 7	174

Gambar 128. Tampilan Menu Tree	180
Gambar 129. Tampilan Menu Insert Tree	181
Gambar 130. Tampilan Menu PreOrder Tree.....	181
Gambar 131. Tampilan Menu InOrder Tree.....	181
Gambar 132. Tampilan Menu PostOrder Tree	182
Gambar 133. Tampilan Menu Exit.....	182
Gambar 134. Ilustrasi Insertion Pada Tree.....	185
Gambar 135 Ilustrasi PreOrder Pada Tree	186
Gambar 136. Ilustrasi InOrde Pada Tree.....	186
Gambar 137. Ilustrasi PostOrder Pada Tree.....	187

DAFTAR TABEL

Tabel 1. Source Code Soal 1 Modul 1.....	13
Tabel 2. Source Code Soal 2 Modul 1.....	19
Tabel 3. Source Code Soal 3 Modul 1.....	24
Tabel 4. Source Code Soal 2 Modul 2.....	30
Tabel 5. Source Code Soal 3 Modul 2.....	42
Tabel 6. Source Code Soal 1 Modul 3.....	55
Tabel 7. Source Code Soal 1 Modul 4.....	95
Tabel 8. Source Code Soal 1 Modul 5.....	126
Tabel 9. Source Code Soal 1 Modul 6.....	157
Tabel 10. Source Code Soal 1 Modul 7.....	175

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";
        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. Soal 1 Modul 1

A. Source Code

Tabel 1. Source Code Soal 1 Modul 1

```
1 #include <iostream>
2
3 using namespace std;
4
5 struct mhs
6 {
7     char nama[20], nim[10], jurusan[2];
8     int sks, program;
9 };
10
11 struct mhs bayar[2];
12
13 int main() {
14     int bts, var, tetap;
15     for(int i=0; i<2; i++)
16     {
17         //input data
18         cout << "\n\n-----\n";
19         cout << "\nNama mhs      = "; cin
20         >> bayar[i].nama;
21         cout << "NIM          = "; cin >>
22         bayar[i].nim;
23         cout << "Jurusan[TI, PTK] = "; cin >>
24         bayar[i].jurusan;
25         cout << "Program[1=D3, 2=S1] = "; cin >>
26         bayar[i].program;
27         input :
28             if      (bayar[i].program < 0 || bayar[i].program > 2)
29             {
```

```

26             cout << "Program tidak sesuai\n";
27             goto input;
28         }     cout << "Jumlah sks           = ";
29         cin >> bayar[i].sks;
30
31         if (bayar[i].program == 1)
32         {
33             tetap = 500000;
34             var = bayar[i].sks*25000;
35
36         else if (bayar[i].program == 2)
37         {
38             tetap = 750000;
39             var = bayar[i].sks*50000;
40         }     cout << endl;
41
42         //Output data
43         cout << "\n\n-----\n-----\n";
44         cout << "Output";
45         cout << "\n-----\n-----\n";
46         cout << "\nNama mhs           = " <<
47             bayar[i].nama;
48         cout << "\nNIM                 = " <<
49             bayar[i].nim;
50         cout << "\nJurusan             = " <<
51             bayar[i].jurusan;
52         cout << "\nProgram             = " <<
53             bayar[i].program;

```

```

50         cout << "\nJumlah sks" = " <<
51         bayar[i].sks;
52         cout << "\nSPP tetap" = " << tetap;
53         cout << "\nSPP variabel" = " << var;
54         cout << endl << endl;
55     }
56 }

```

B. Output Program

```

task-1-struct-and-pointer-Kelompok-A-M
PS D:\task-1-struct-and-pointer-Kelompok-A-M> cd "D:\task-1-struct-and-pointer-Kelompok-A-M"
PS D:\task-1-struct-and-pointer-Kelompok-A-M> g++ Problem.cpp -o Problem
PS D:\task-1-struct-and-pointer-Kelompok-A-M> ./Problem
-----[Output]-----
Name mhs = Adit
IDN = 24388171
Jurusan = TI
Program = TI
Jumlah sks = 12
SPP tetap = 120000
SPP variabel = 450000

-----[Output]-----
Name mhs = Rizki
IDN = 24388171
Jurusan = TI
Program = TI
Jumlah sks = 12
SPP tetap = 120000
SPP variabel = 450000

-----[Output]-----
Name mhs = Arifan
IDN = 24388171
Jurusan = TI
Program = TI
Jumlah sks = 12
SPP tetap = 120000
SPP variabel = 450000

```

Gambar 2. Screenshot Output Program Soal 1 Modul 1

C. Pembahasan

Pada baris [1] terdapat `#include <iostream>` yang mana merupakan komponen pertama dan terpenting untuk sebuah kode pada bahasa pemrograman cpp untuk mengakses library standard input dan output file. `#Include` sendiri merupakan kode pada bahasa pemrograman cpp yang digunakan untuk mengakses sebuah file yang diinginkan. Sementara itu `Iostream` merupakan sebuah file library yang berisi fungsi standar input dan output yang ada pada bahasa pemrograman cpp.

Pada baris [3] terdapat `using namespace std;` yang mana digunakan untuk menghindari penulisan `std::` untuk setiap penggunaan fungsi seperti `cin` dan `cout` yang ada.

Pada baris [5] sampai [9] terdapat `struct mhs` yang mana merupakan sebuah struktur yang dibuat untuk menampung lima jenis data berbeda yang diinginkan

sekaligus untuk keperluan program ini. Data yang ditampung dalam bentuk *Char* untuk *nama, nim, dan jurusan; Int untuk sks dan program.*

Pada baris [7] sampai [50] terdapat *[n]* yang mana merupakan array. Array digunakan disini sebagai pembatas atau banyaknya nilai atau data yang dapat diinput kedalam variabel yang ada, banyaknya nilai data atau nilai yang ditampung semua tergantung dengan nilai yang ada di array atau yang telah ditentukan sebelumnya.

Pada baris [6] sampai [56] terdapat *{ }* yang mana sepasang tanda kurung kurawal digunakan untuk menandai awal dan akhir serangkaian pernyataan. Oleh karena itu semua fungsi harus dimulai dan diakhiri dengan tanda kurung kurawal.

Pada baris [11] terdapat *Struct mhs bayar[2]* yang mana merupakan sebuah variabel dari struct yang diberi nama bayar yang kedepannya akan dibagi dalam *bayar[i].nama, bayar[i].nim, bayar[i].jurusan, bayar[i].program, dan bayar[i].sks.* setelah dibagi dalam kelima bentuk, masing-masing bentuk akan memegang atau menyimpan nilai atau data yang di inputkan. Array yang ada pada *Struct mhs bayar[2]*, akan menampung data untuk dua orang mahasiswa jadi setelah selesai melakukan penginputan pertama akan muncul penginputan kedua untuk mahasiswa.

Pada baris [13] terdapat *int main()* yang mana sebagai titik awal eksekusi program. *int* yang terdapat di depan *main* adalah jenis tipe data yang akan dikembalikan nilainya kembali. Di dalam fungsi *main()*, wajib untuk disertakan *return 0*. Artinya, fungsi main yang ada akan mengembalikan nilai 0 setelah dieksekusi.

Pada baris [14] terdapat *int bts, var, dan tetap* yang mana digunakan untuk menampilkan atau menyimpan tipe data dalam bentuk integer atau bilangan bulat.

Pada baris [15] terdapat *for(int i=0; i<2; i++)* yang mana digunakan sebagai perulangan atau loop untuk proses input output yang ada.\

Pada baris [18] sampai [52] terdapat */n* yang mana digunakan ketika ingin membuat new line atau baris baru pada tampilan output.

Pada baris [18] sampai [53] terdapat *cout* yang mana berfungsi untuk menampilkan output ke layar komputer. *cout* ketika diisi dengan string atau kalimat

akan diawali dan diakhiri tanda *petik ganda* ("") dan selalu di akhiri dengan *titik koma* (;). Terus terdapatnya variable seperti *data.Huruf*, *data.Kata*, dan *data.Angka* di akhir tanda petik dua pada *cout* adalah untuk mencetak nilai yang terdapat pada variable tersebut pada kalimat.

Pada baris [19] sampai [22] terdapat *cin* yang mana berfungsi untuk pengimputan nilai atau data yang kita inginkan ke variable dalam bentuk integer, float atau char sesuai dengan bentuk atau jenis variable yang di panggil atau yang telah ditentukan sebelumnya.

Pada baris [24] sampai [28] terdapat *if (bayar[i].program < 0 || bayar[i].program > 2)* yang mana merupakan bagian *if-else* dari program. Untuk mengatur inputan yang masuk ke dalam program, jika inputan yang dimasukkan pada program di luar 1 atau 2, maka program yang ada akan mencetak pesan error dan kembali meminta input kepada user.

Pada baris [30] sampai [34] terdapat *if (bayar[i].program == 1)* yang mana merupakan sambungan dari *if-else* dari program sebelumnya. Setelah berhasi memasukkan inputan yang benar, penjumlahan yang ada akan menghitung banyaknya yang harus dibayar sesuai dengan program yang dipilih dan disini milih 1 untuk D3 yang mempunyai *tetap* sebanyak Rp500.000 dan menghitung *var* yang ada dengan mengali banyaknya sks yang diinput dengan Rp25.000.

Pada baris [36] sampai [40] terdapat *else if (bayar[i].program == 2)* yang mana merupakan sambungan dari *if-else* dari program sebelumnya. Setelah berhasi memasukkan inputan yang benar, penjumlahan yang ada akan menghitung banyaknya yang harus dibayar sesuai dengan program yang dipilih dan disini milih 2 untuk S1 yang mempunyai *tetap* sebanyak Rp750.000 dan menghitung *var* yang ada dengan mengali banyaknya sks yang diinput dengan Rp50.000.

Pada baris [40] dan [53] terdapat *endl* yang mana digunakan untuk mengakhiri sebuah lane dan membuat lane baru yang dapat digunakan untuk melanjutkan program yang ada.

Pada baris [55] terdapat *return 0*; yang mana mengacu pada nilai-nilai yang dikembalikan dari suatu fungsi. *return* dalam program kita mengembalikan nilai

dari *main()*. Nilai yang dikembalikan oleh main menunjukkan status penghentian program.

SOAL 2

Buatlah program dengan menggunakan struct dengan hasil eksekusi program sebagai berikut:

- a. Plat Nomor Kendaraan : DA1234MK
- b. Jenis Kendaraan : RUSH
- c. Nama Pemilik : Andika Hartanto
- d. Alamat : Jl. Kayu Tangi 1
- e. Kota : Banjarmasin

A. Source Code

Tabel 2. Source Code Soal 2 Modul 1

```
1 #include<iostream>
2
3 using namespace std;
4
5 struct Kendaraan
6 {
7     char Plat_Nomor[15];
8     char Jenis_Kendaraan[30];
9     char Nama_Pemilik[50];
10    char Alamat[100];
11    char Kota[30];
12 };
13
14 int main() {
15     Kendaraan data;
16
17     //Input Data
18     cout << "\n-----\n";
19     cout << "a. Plat Nomor Kendaraan : ";
20     cin.getline(data.Plat_Nomor, 15);
```

```

21     cout << "b. Jenis Kendaraan      : ";
22     cin.getline(data.Jenis_Kendaraan, 30);
23     cout << "c. Nama Pemilik       : ";
24     cin.getline(data.Nama_Pemilik, 50);
25     cout << "d. Alamat             : ";
26     cin.getline(data.Alamat, 100);
27     cout << "e. Kota                : ";
28     cin.getline(data.Kota, 30);
29     cout << "-----\n\n";
30
31     //Output Data
32     cout << "-----\n";
33     cout << "a. Plat Nomor Kendaraan : " <<
data.Plat_Nomor << endl;
34     cout << "b. Jenis Kendaraan      : " <<
data.Jenis_Kendaraan << endl;
35     cout << "c. Nama Pemilik       : " <<
data.Nama_Pemilik << endl;
36     cout << "d. Alamat             : " <<
data.Alamat << endl;
37     cout << "e. Kota                : " << data.Kota
<< endl;
38     cout << "-----\n";
39
40     return 0;
41 }
```

B. Output Program

```
File Edit Selection View ... ↵ task-1-struct-and-pointer-Riki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

EXPLORER ...
TAKS-1-STRUCT-AND-POINTER ...
MaterialsAndTools ...
Problem1.cpp M
Problem2.cpp M
Problem3.cpp M
Problem4.cpp M
Problem5.cpp M
Problem6.cpp M
Problem7.cpp M
Problem8.cpp M
Problem9.cpp M
Problem10.cpp M
Problem11.cpp M
Problem12.cpp M
Problem13.cpp M
Problem14.cpp M
Problem15.cpp M
Problem16.cpp M
Problem17.cpp M
Problem18.cpp M
Problem19.cpp M
Problem20.cpp M
Problem21.cpp M
Problem22.cpp M
Problem23.cpp M
Problem24.cpp M
Problem25.cpp M
Problem26.cpp M
Problem27.cpp M
Problem28.cpp M
Problem29.cpp M
Problem30.cpp M
Problem31.cpp M
Problem32.cpp M
Problem33.cpp M
Problem34.cpp M
Problem35.cpp M
Problem36.cpp M
Problem37.cpp M
Problem38.cpp M
Problem39.cpp M
Problem40.cpp M
Problem41.cpp M
Problem42.cpp M
Problem43.cpp M
Problem44.cpp M
Problem45.cpp M
Problem46.cpp M
Problem47.cpp M
Problem48.cpp M
Problem49.cpp M
Problem50.cpp M
Problem51.cpp M
Problem52.cpp M
Problem53.cpp M
Problem54.cpp M
Problem55.cpp M
Problem56.cpp M
Problem57.cpp M
Problem58.cpp M
Problem59.cpp M
Problem60.cpp M
Problem61.cpp M
Problem62.cpp M
Problem63.cpp M
Problem64.cpp M
Problem65.cpp M
Problem66.cpp M
Problem67.cpp M
Problem68.cpp M
Problem69.cpp M
Problem70.cpp M
Problem71.cpp M
Problem72.cpp M
Problem73.cpp M
Problem74.cpp M
Problem75.cpp M
Problem76.cpp M
Problem77.cpp M
Problem78.cpp M
Problem79.cpp M
Problem80.cpp M
Problem81.cpp M
Problem82.cpp M
Problem83.cpp M
Problem84.cpp M
Problem85.cpp M
Problem86.cpp M
Problem87.cpp M
Problem88.cpp M
Problem89.cpp M
Problem90.cpp M
Problem91.cpp M
Problem92.cpp M
Problem93.cpp M
Problem94.cpp M
Problem95.cpp M
Problem96.cpp M
Problem97.cpp M
Problem98.cpp M
Problem99.cpp M
Problem100.cpp M
Problem101.cpp M
Problem102.cpp M
Problem103.cpp M
Problem104.cpp M
Problem105.cpp M
Problem106.cpp M
Problem107.cpp M
Problem108.cpp M
Problem109.cpp M
Problem110.cpp M
Problem111.cpp M
Problem112.cpp M
Problem113.cpp M
Problem114.cpp M
Problem115.cpp M
Problem116.cpp M
Problem117.cpp M
Problem118.cpp M
Problem119.cpp M
Problem120.cpp M
Problem121.cpp M
Problem122.cpp M
Problem123.cpp M
Problem124.cpp M
Problem125.cpp M
Problem126.cpp M
Problem127.cpp M
Problem128.cpp M
Problem129.cpp M
Problem130.cpp M
Problem131.cpp M
Problem132.cpp M
Problem133.cpp M
Problem134.cpp M
Problem135.cpp M
Problem136.cpp M
Problem137.cpp M
Problem138.cpp M
Problem139.cpp M
Problem140.cpp M
Problem141.cpp M
Problem142.cpp M
Problem143.cpp M
Problem144.cpp M
Problem145.cpp M
Problem146.cpp M
Problem147.cpp M
Problem148.cpp M
Problem149.cpp M
Problem150.cpp M
Problem151.cpp M
Problem152.cpp M
Problem153.cpp M
Problem154.cpp M
Problem155.cpp M
Problem156.cpp M
Problem157.cpp M
Problem158.cpp M
Problem159.cpp M
Problem160.cpp M
Problem161.cpp M
Problem162.cpp M
Problem163.cpp M
Problem164.cpp M
Problem165.cpp M
Problem166.cpp M
Problem167.cpp M
Problem168.cpp M
Problem169.cpp M
Problem170.cpp M
Problem171.cpp M
Problem172.cpp M
Problem173.cpp M
Problem174.cpp M
Problem175.cpp M
Problem176.cpp M
Problem177.cpp M
Problem178.cpp M
Problem179.cpp M
Problem180.cpp M
Problem181.cpp M
Problem182.cpp M
Problem183.cpp M
Problem184.cpp M
Problem185.cpp M
Problem186.cpp M
Problem187.cpp M
Problem188.cpp M
Problem189.cpp M
Problem190.cpp M
Problem191.cpp M
Problem192.cpp M
Problem193.cpp M
Problem194.cpp M
Problem195.cpp M
Problem196.cpp M
Problem197.cpp M
Problem198.cpp M
Problem199.cpp M
Problem200.cpp M
Problem201.cpp M
Problem202.cpp M
Problem203.cpp M
Problem204.cpp M
Problem205.cpp M
Problem206.cpp M
Problem207.cpp M
Problem208.cpp M
Problem209.cpp M
Problem210.cpp M
Problem211.cpp M
Problem212.cpp M
Problem213.cpp M
Problem214.cpp M
Problem215.cpp M
Problem216.cpp M
Problem217.cpp M
Problem218.cpp M
Problem219.cpp M
Problem220.cpp M
Problem221.cpp M
Problem222.cpp M
Problem223.cpp M
Problem224.cpp M
Problem225.cpp M
Problem226.cpp M
Problem227.cpp M
Problem228.cpp M
Problem229.cpp M
Problem230.cpp M
Problem231.cpp M
Problem232.cpp M
Problem233.cpp M
Problem234.cpp M
Problem235.cpp M
Problem236.cpp M
Problem237.cpp M
Problem238.cpp M
Problem239.cpp M
Problem240.cpp M
Problem241.cpp M
Problem242.cpp M
Problem243.cpp M
Problem244.cpp M
Problem245.cpp M
Problem246.cpp M
Problem247.cpp M
Problem248.cpp M
Problem249.cpp M
Problem250.cpp M
Problem251.cpp M
Problem252.cpp M
Problem253.cpp M
Problem254.cpp M
Problem255.cpp M
Problem256.cpp M
Problem257.cpp M
Problem258.cpp M
Problem259.cpp M
Problem260.cpp M
Problem261.cpp M
Problem262.cpp M
Problem263.cpp M
Problem264.cpp M
Problem265.cpp M
Problem266.cpp M
Problem267.cpp M
Problem268.cpp M
Problem269.cpp M
Problem270.cpp M
Problem271.cpp M
Problem272.cpp M
Problem273.cpp M
Problem274.cpp M
Problem275.cpp M
Problem276.cpp M
Problem277.cpp M
Problem278.cpp M
Problem279.cpp M
Problem280.cpp M
Problem281.cpp M
Problem282.cpp M
Problem283.cpp M
Problem284.cpp M
Problem285.cpp M
Problem286.cpp M
Problem287.cpp M
Problem288.cpp M
Problem289.cpp M
Problem290.cpp M
Problem291.cpp M
Problem292.cpp M
Problem293.cpp M
Problem294.cpp M
Problem295.cpp M
Problem296.cpp M
Problem297.cpp M
Problem298.cpp M
Problem299.cpp M
Problem299.cpp M
PS D:\task-1-struct-and-pointer-Riki-A-M>
```

Gambar 3. Screenshot Output Program Soal 2 Modul 1

C. Pembahasan

Pada baris [1] terdapat `#include <iostream>` yang mana merupakan komponen pertama dan terpenting untuk sebuah kode pada bahasa pemrograman cpp untuk mengakses library standard input dan output file. `#Include` sendiri merupakan kode pada bahasa pemrograman cpp yang digunakan untuk mengakses sebuah file yang diinginkan. Sementara itu `Iostream` merupakan sebuah file library yang berisi fungsi standar input dan output yang ada pada bahasa pemrograman cpp.

Pada baris [3] terdapat `using namespace std;` yang mana digunakan untuk menghindari penulisan `std::` untuk setiap penggunaan fungsi seperti `cin` dan `cout` yang ada.

Pada baris [5] sampai [12] terdapat *struct Kendaraan* yang mana merupakan sebuah struktur yang dibuat untuk menampung lima jenis data berbeda yang diinginkan sekaligus untuk keperluan program ini. Data yang ditampung dalam bentuk *Char* untuk *Plat_Nomor, Jenis_Kendaraan, Nama_Pemilik, Alamat, dan Kota*.

Pada baris [7] sampai [11] terdapat `[n]` yang mana merupakan array. Array digunakan disini sebagai pembatas atau banyaknya nilai atau data yang dapat diinput kedalam variabel yang ada, banyaknya nilai data atau nilai yang ditampung semua tergantung dengan nilai yang ada di array atau yang telah ditentukan sebelumnya.

Pada baris [6] sampai [12] dan [14] sampai [41] terdapat `{ }` yang mana sepasang tanda kurung kurawal digunakan untuk menandai awal dan akhir serangkaian pernyataan. Oleh karena itu semua fungsi harus dimulai dan diakhiri dengan tanda kurung kurawal.

Pada baris [14] terdapat `int main()` yang mana sebagai titik awal eksekusi program. `int` yang terdapat di depan `main` adalah jenis tipe data yang akan dikembalikan nilainya kembali. Di dalam fungsi `main()`, wajib untuk disertakan `return 0`. Artinya, fungsi `main` yang ada akan mengembalikan nilai 0 setelah dieksekusi.

Pada baris [15] terdapat `Inputan` data yang mana merupakan sebuah variabel dari `struct` yang diberi nama `data` yang kedepannya akan dibagi dalam `data.Plat_Nomor`, `data.Jenis_Kendaraan`, `data.Nama_Pemilik`, `data.Alamat`, dan `data.Kota`. setelah dibagi dalam kelima bentuk, masing-masing bentuk akan memegang atau menyimpan nilai atau data yang di inputkan

Pada baris [18] sampai [38] terdapat `/n` yang mana digunakan ketika ingin membuat new lane atau baris baru pada tampilan output.

Pada baris [18] sampai [38] terdapat `cout` yang mana berfungsi untuk menampilkan output ke layar komputer. `cout` ketika diisi dengan string atau kalimat akan diawali dan diakhiri tanda petik ganda ("") dan selalu di akhiri dengan titik koma (;). Terus terdapatnya variable seperti `data.Huruf`, `data.Kata`, dan `data.Angka` di akhir tanda petik dua pada `cout` adalah untuk mencetak nilai yang terdapat pada variable tersebut pada kalimat.

Pada baris [20] sampai [28] terdapat `cin` yang mana berfungsi untuk pengimputan nilai atau data yang kita inginkan ke variable dalam bentuk `integer`, `float` atau `char` sesuai dengan bentuk atau jenis variable yang di panggil atau yang telah ditentukan sebelumnya. `cin.getline()` digunakan agar spasi yang ada ketika pengimputan dapat terbaca, sehingga program yang ada dapat berjalan dengan semestinya.

Pada baris [33] sampai [37] terdapat `endl` yang mana digunakan untuk mengakhiri sebuah lane dan membuat lane baru yang dapat digunakan untuk melanjutkan program yang ada.

Pada baris [40] terdapat `return 0;` yang mana mengacu pada nilai-nilai yang dikembalikan dari suatu fungsi. `return` dalam program kita mengembalikan nilai dari `main()`. Nilai yang dikembalikan oleh `main` menunjukkan status penghentian program.

SOAL 3

Buatlah program dengan tampilan sebagai berikut:

- a. Masukkan sebuah huruf =
- b. Masukan sebuah kata =
- c. Masukkan Angka =
- d. Huruf yang Anda masukkan adalah
- e. Kata yang Anda masukkan adalah
- f. Angka yang Anda masukkan adalah

A. Source Code

Tabel 3. Source Code Soal 3 Modul 1

```
1 #include<iostream>
2
3 using namespace std;
4
5 struct Inputan
6 {
7     char Huruf;
8     string Kata;
9     int Angka;
10};
11
12 int main(){
13     Inputan data;
14
15     //Input Data
16     cout << "\n-----\n";
17     cout << "a. Masukkan Sebuah huruf = ";
18     cin >> data.Huruf;
19     cout << "b. Masukkan Sebuah kata = ";
20     cin >> data.Kata;
```

```

21     cout << "c. Masukkan Angka = ";
22     cin >> data.Angka;
23
24     //Output Data
25     cout << "d. Huruf yang Anda masukkan adalah "
26     << data.Huruf << endl;
27     cout << "e. Kata yang Anda masukkan adalah "
28     << data.Kata << endl;
29     cout << "f. Angka yang Anda masukkan adalah "
30     << data.Angka << endl;
31
32     cout << "-----\n";
33
34     return 0;
35 }
```

B. Output Program

```

File Edit Selection View ... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS task 1 struct and pointer Rizki A M
EXPLORER ...
TASK-1-STRUCT-AND-PO... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\task-1-struct-and-pointer-Rizki-A-10 cd "D:\task-1-struct-and-pointer-Rizki-A-10" ; if ($?) { g++ Problem1.cpp -o Problem1 } ; if ($?) { .\Problem1 }
a. Masukkan sebuah huruf - N
b. Masukkan sebuah kata - Nimeroku
c. Masukkan Angka - 20
d. Huruf yang Anda masukkan adalah N
e. Kata yang Anda masukkan adalah Nimeroku
f. Angka yang Anda masukkan adalah 20
PS D:\task-1-struct-and-pointer-Rizki-A-10

```

Gambar 4 Screenshot Output Program Soal 3 Modul 1

C. Pembahasan

Pada baris [1] terdapat `#include <iostream>` yang mana merupakan komponen pertama dan terpenting untuk sebuah kode pada bahasa pemrograman cpp untuk mengakses library standard input dan output file. `#Include` sendiri merupakan kode pada bahasa pemrograman cpp yang digunakan untuk mengakses sebuah file yang diinginkan. Sementara itu `Iostream` merupakan sebuah file

library yang berisi fungsi standar input dan output yang ada pada bahasa pemrograman cpp.

Pada baris [3] terdapat `using namespace std;` yang mana digunakan untuk menghindari penulisan `std::` untuk setiap penggunaan fungsi seperti `cin` dan `cout` yang ada.

Pada baris [5] sampai [10] terdapat `struct Inputan` yang mana merupakan sebuah struktur yang dibuat untuk menampung tiga jenis data berbeda yang diinginkan sekaligus untuk keperluan program ini. Data yang ditampung dalam bentuk `Char` untuk *Huruf*, `string` untuk *Kata*, dan `integer` untuk *Angka*.

Pada baris [6] sampai [10] dan [12] sampai [30] terdapat `{ }` yang mana sepasang tanda kurung kurawal digunakan untuk menandai awal dan akhir serangkaian pernyataan. Oleh karena itu semua fungsi harus dimulai dan diakhiri dengan tanda kurung kurawal.

Pada baris [12] terdapat `int main()` yang mana sebagai titik awal eksekusi program. `int` yang terdapat di depan `main` adalah jenis tipe data yang akan dikembalikan nilainya kembali. Di dalam fungsi `main()`, wajib untuk disertakan `return 0`. Artinya, fungsi main yang ada akan mengembalikan nilai 0 setelah dieksekusi.

Pada baris [13] terdapat `Inputan data` yang mana merupakan sebuah variabel dari `struct` yang diberi nama `data` yang kedepannya akan dibagi dalam `data.Huruf`, `data.Kata`, dan `data.Angka`. setelah dibagi dalam ketiga bentuk, masing-masing bentuk akan memegang atau menyimpan nilai atau data yang di inputkan ke dalamnya ketika diinput atau diisi.

Pada baris [16] dan [28] terdapat `/n` yang mana digunakan ketika ingin membuat new line atau baris baru pada tampilan output.

Pada baris [16] sampai [28] terdapat `cout` yang mana berfungsi untuk menampilkan output ke layar komputer. `cout` ketika diisi dengan string atau kalimat akan diawali dan diakhiri tanda *petik ganda* ("") dan selalu di akhiri dengan *titik koma* (;). Terus terdapatnya variable seperti `data.Huruf`, `data.Kata`, dan `data.Angka` di akhir tanda petik dua pada `cout` adalah untuk mencetak nilai yang terdapat pada variable tersebut pada kalimat.

Pada baris [18] sampai [22] terdapat `cin` yang mana berfungsi untuk pengimputan nilai atau data yang kita inginkan ke variable dalam bentuk integer, float atau char sesuai dengan bentuk atau jenis variable yang di panggil atau yang telah ditentukan sebelumnya.

Pada baris [25] sampai [27] terdapat `endl` yang mana digunakan untuk mengakhiri sebuah lane dan membuat lane baru yang dapat digunakan untuk melanjutkan program yang ada.

Pada baris [29] terdapat `return 0;` yang mana mengacu pada nilai-nilai yang dikembalikan dari suatu fungsi. `return` dalam program kita mengembalikan nilai dari `main()`. Nilai yang dikembalikan oleh main menunjukkan status penghentian program.

MODUL 2 : STACK DAN QUEUE

SOAL 1

Apa perbedaan Stack dengan Queue?

A. Pembahasan

Stack dan Queue adalah struktur data yang mempunyai perbedaan dalam cara memasukkan/input (*push/enqueue*) dan mengeluarkan/output (*pop/dequeue*) data atau elemen yang ada.

Pada struktur data Stack, prinsip yang digunakan adalah **LIFO** (*Last in First Out*), yang mana data atau elemen yang paling terakhir dimasukkan (*push*) akan menjadi data atau elemen yang paling awal dikeluarkan (*pop*). Sebagai contoh, kita ambil analogi seperti ketika kita menumpuk piring kotor yang ada, piring kotor yang berada di tumpukkan paling bawah akan selesai dibersihkan paling akhir sedangkan, piring kotor yang terakhir ditumpuk akan menjadi yang paling awal untuk dibersihkan.

Sedangkan, pada struktur data Queue, prinsip yang digunakan adalah **FIFO** (*First in First Out*), yang mana data atau elemen yang pertama dimasukkan (*enqueue*) akan menjadi data atau elemen yang pertama untuk dikeluarkan (*dequeue*). Sebagai contoh kita ambil analogi seperti ketika kita mengantri untuk memesan gacoan, orang yang datang atau mendaftar pertama kali ke meja kasih ialah yang akan dilayani lebih dulu.

SOAL 2

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

```
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 5 Soal 2 Modul 2

A. Source Code

Tabel 4. Source Code Soal 2 Modul 2

```
1 #include <iostream>
2 #include <conio.h>
3 #include <stdlib.h>
4
5 #define MAX 20
6
7 using namespace std;
8
9 struct Stack
10 {
11     int Atas;
12     int Data[MAX];
13 };
14
15 Stack Tumpuk;
16
17 int Kosong()
18 {
19     if (Tumpuk. Atas == -1)
20     {
21         return 1;
22     }
23     else
24     {
25         return 0;
26     }
27 }
28
29 int Penuh()
30 {
31     if (Tumpuk. Atas == MAX-1)
```

```

32    {
33        return 1;
34    }
35    else
36    {
37        return 0;
38    }
39 }
40
41 void Push(int Data)
42 {
43     if (Kosong ()==1)
44     {
45         Tumpuk.Atas++;
46         Tumpuk.Data [Tumpuk.Atas] = Data;
47         cout << "Data " <<
Tumpuk.Data [Tumpuk.Atas] << " dimasukkan ke dalam
Stack." << endl;
48     }
49     else if (Penuh ()==0)
50     {
51         Tumpuk.Atas++;
52         Tumpuk.Data [Tumpuk.Atas] = Data;
53         cout << "Data " <<
Tumpuk.Data [Tumpuk.Atas] << " dimasukkan ke dalam
Stack." << endl;
54     }
55     else
56     {
57         cout << "Stack telah Penuh!!!" << endl;
58     }
59 }

```

```

60
61 void Pop()
62 {
63     if (Kosong() == 0)
64     {
65         cout << "Data " <<
66         Tumpuk.Data[Tumpuk.Atas] << " diambil dari
67         Stack." << endl;
68         Tumpuk.Atas--;
69     }
70     else
71     {
72         cout << "Stack Kosong!!!" << endl;
73     }
74 }
75 void Cetak_Stack()
76 {
77     if (Kosong() == 0)
78     {
79         for (int i = Tumpuk.Atas; i >= 0; i--)
80         {
81             cout << "\nTumpukan Ke " << i << " =
82             " << Tumpuk.Data[i];
83         }
84         cout << endl;
85     }
86     else
87     {
88         cout << "Stack Kosong!!!" << endl;
89     }
90 }
```

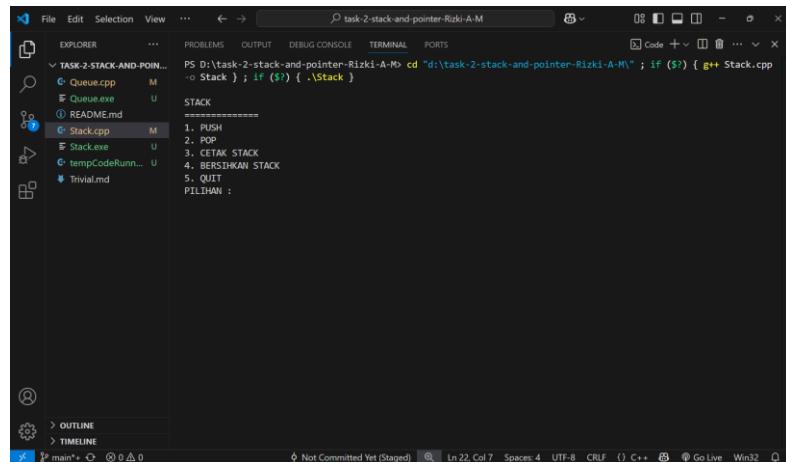
```

89
90 void Bersihkan_Stack()
91 {
92     Tumpuk.Atas = -1;
93     cout << "Stack telah dibersihkan!!!" << endl;
94 }
95
96 void Inisialisasi()
97 {
98     Tumpuk.Atas = -1;
99 }
100
101 int main()
102 {
103     Inisialisasi();
104     int Pilihan, Data;
105     do{
106         cout << "\nSTACK" << endl;
107         cout << "======" << endl;
108         cout << "1. PUSH" << endl;
109         cout << "2. POP" << endl;
110         cout << "3. CETAK STACK" << endl;
111         cout << "4. BERSIHKAN STACK" << endl;
112         cout << "5. QUIT" << endl;
113         cout << "PILIHAN : "; cin >> Pilihan;
114         switch (Pilihan)
115         {
116             case 1:
117                 cout << "Masukkan Nilai: "; cin >>
Data;
118                 Push(Data);
119                 break;

```

```
120
121     case 2:
122         Pop();
123         break;
124
125     case 3:
126         Cetak_Stack();
127         break;
128
129     case 4:
130         Bersihkan_Stack();
131         break;
132
133     default:
134         cout << "TERIMA KASIH" << endl;
135         break;
136     }
137     cout << "Press any key to continue";
138     getch();
139     system("cls");
140 }
141 while (Pilihan < 5);
142 return 0;
143 }
```

B. Output Program

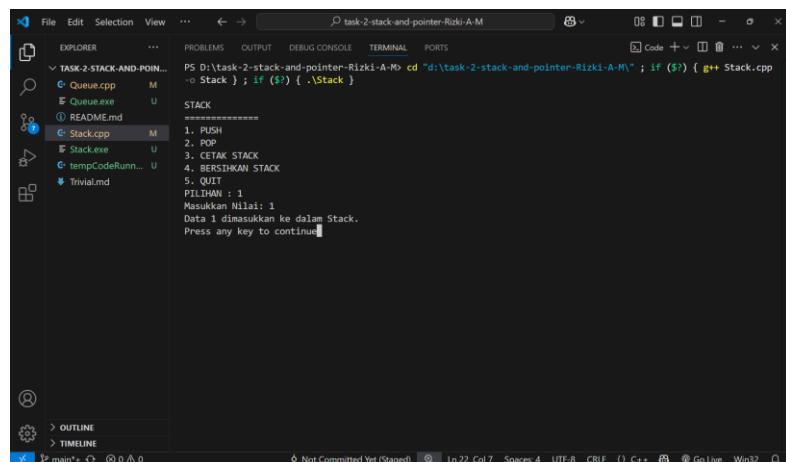


VS Code interface showing the project structure in the Explorer panel and the terminal output in the Terminal panel. The terminal shows the program's menu and a prompt for input.

```
PS D:\task-2-stack-and-pointer-Rizki-A-M> cd "d:\task-2-stack-and-pointer-Rizki-A-M" ; IF ($?) { g++ Stack.cpp
-> Stack } ; IF ($?) { .\Stack }

STACK
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN :
```

Gambar 6. Tampilan Awal Program Saat Dijalankan

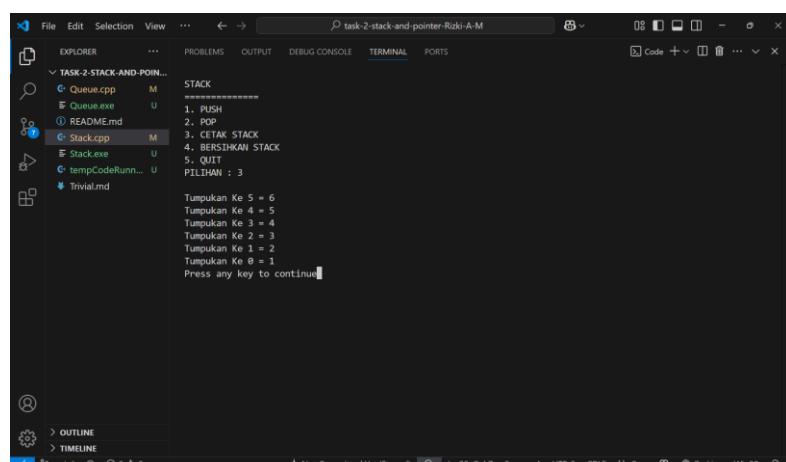


VS Code interface showing the project structure in the Explorer panel and the terminal output in the Terminal panel. The terminal shows the program's menu and a prompt for input.

```
PS D:\task-2-stack-and-pointer-Rizki-A-M> cd "d:\task-2-stack-and-pointer-Rizki-A-M" ; IF ($?) { g++ Stack.cpp
-> Stack } ; IF ($?) { .\Stack }

STACK
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 1
Masukkan Nilai: 1
Data 1 dimasukkan ke dalam Stack.
Press any key to continue
```

Gambar 7. Memasukkan Nilai Ke Dalam Stack

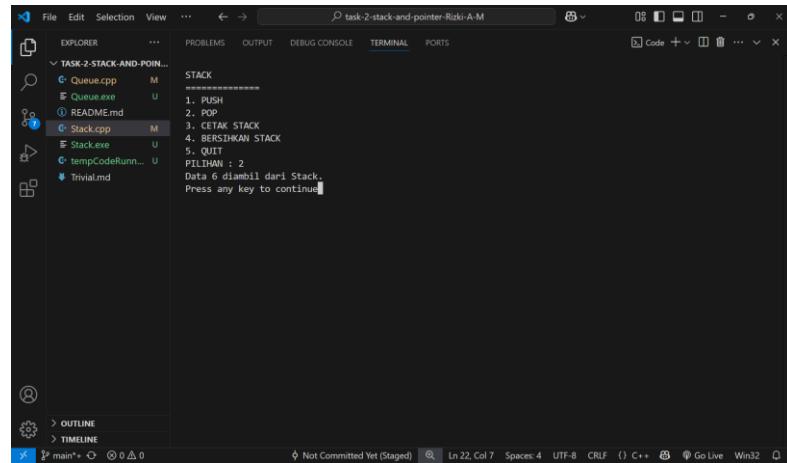


VS Code interface showing the project structure in the Explorer panel and the terminal output in the Terminal panel. The terminal shows the program's menu, a push operation, and the resulting stack values.

```
PS D:\task-2-stack-and-pointer-Rizki-A-M> cd "d:\task-2-stack-and-pointer-Rizki-A-M" ; IF ($?) { g++ Stack.cpp
-> Stack } ; IF ($?) { .\Stack }

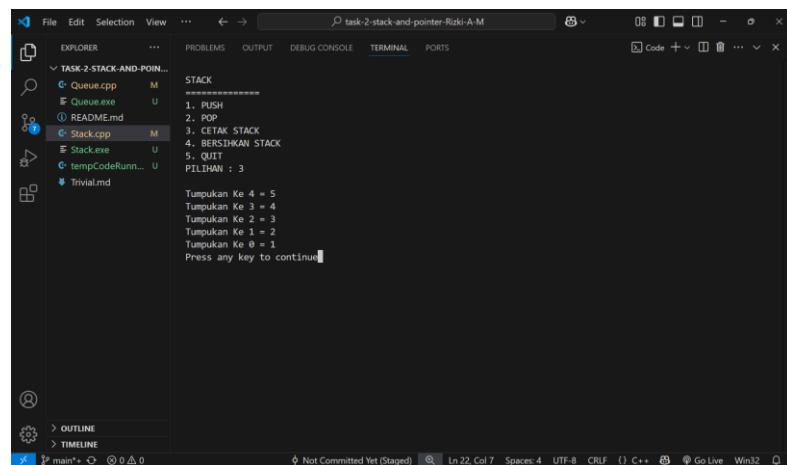
STACK
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 3
Tumpukan Ke 5 = 6
Tumpukan Ke 4 = 5
Tumpukan Ke 3 = 4
Tumpukan Ke 2 = 3
Tumpukan Ke 1 = 2
Tumpukan Ke 0 = 1
Press any key to continue
```

Gambar 8. Menampilkan Stack Yang Sudah Dimasukkan



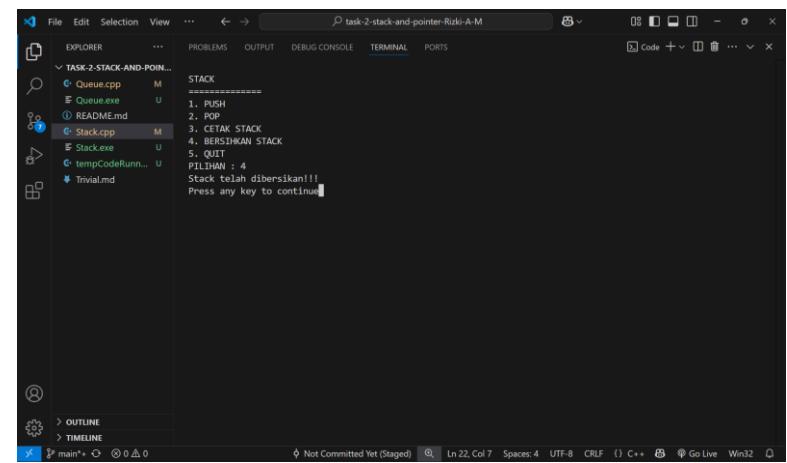
```
task-2-stack-and-pointer-Rizki-A-M
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TASK-2-STACK-AND-POIN...
Queue.cpp M
Queue.exe U
README.md
Stack.cpp M
Stack.exe U
tempCodeRunn... U
Trivial.md
STACK
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 2
Data 6 diambil dari Stack.
Press any key to continue
```

Gambar 9. Melakukan Pop Pada Nilai Di Dalam Stack



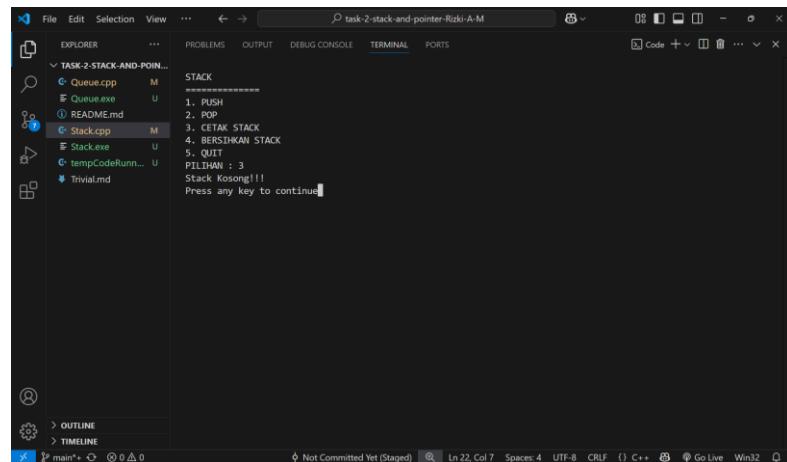
```
task-2-stack-and-pointer-Rizki-A-M
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TASK-2-STACK-AND-POIN...
Queue.cpp M
Queue.exe U
README.md
Stack.cpp M
Stack.exe U
tempCodeRunn... U
Trivial.md
STACK
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 3
Tumpukan Ke 4 = 5
Tumpukan Ke 3 = 4
Tumpukan Ke 2 = 3
Tumpukan Ke 1 = 2
Tumpukan Ke 0 = 1
Press any key to continue
```

Gambar 10. Tampilan Setelah Nilai Teratas Di Pop



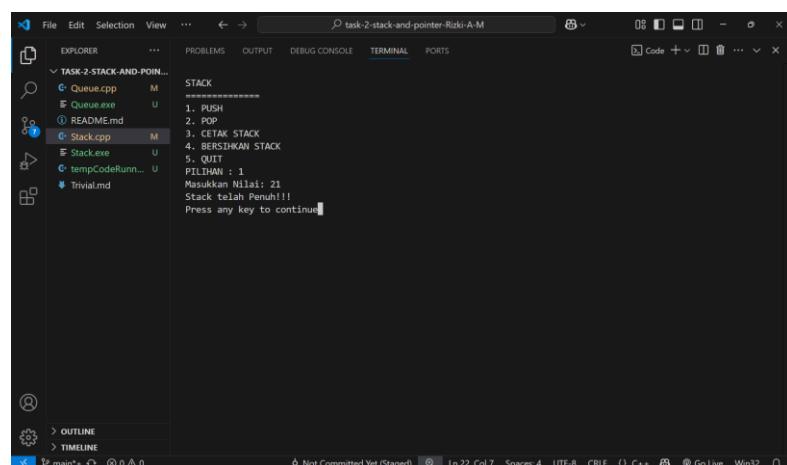
```
task-2-stack-and-pointer-Rizki-A-M
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TASK-2-STACK-AND-POIN...
Queue.cpp M
Queue.exe U
README.md
Stack.cpp M
Stack.exe U
tempCodeRunn... U
Trivial.md
STACK
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 4
Stack telah dibersihkan!!
Press any key to continue
```

Gambar 11. Membersihkan Stack Dengan Pilihan Menu 4



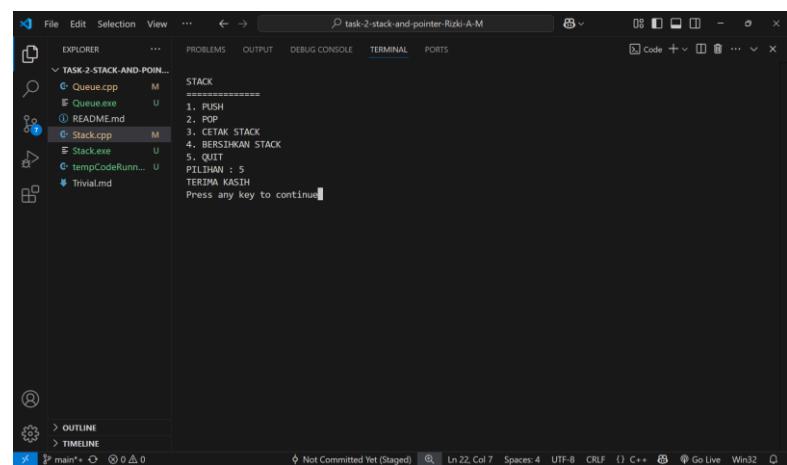
```
task-2-stack-and-pointer-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
EXPLORER TASK-2-STACK-AND-POIN...
Queue.cpp M Queue.exe U README.md
Stack.cpp M Stack.exe U tempCodeRunn... U Trivial.lmd
Stack
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHAN STACK
5. QUIT
PILIHAN : 3
Masukkan Nilai: 21
Stack Kosong!!!
Press any key to continue
```

Gambar 12. Tampilan Setelah Dibersihkan



```
task-2-stack-and-pointer-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
EXPLORER TASK-2-STACK-AND-POIN...
Queue.cpp M Queue.exe U README.md
Stack.cpp M Stack.exe U tempCodeRunn... U Trivial.lmd
Stack
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHAN STACK
5. QUIT
PILIHAN : 1
Masukkan Nilai: 21
Stack telah Penuh!!!
Press any key to continue
```

Gambar 13. Tampilan Jika Stack Sudah Penuh



```
task-2-stack-and-pointer-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
EXPLORER TASK-2-STACK-AND-POIN...
Queue.cpp M Queue.exe U README.md
Stack.cpp M Stack.exe U tempCodeRunn... U Trivial.lmd
Stack
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHAN STACK
5. QUIT
PILIHAN : 5
TERIMA KASIH
Press any key to continue
```

Gambar 14. Tampilan Saat Memilih Menu 5 Quit

C. Pembahasan

Pada baris [1] sampai [3] terdapat `#include` yang mana digunakan untuk mengakses sebuah file yang diinginkan. `<iostream>` yang ada digunakan untuk input dan output. Kemudian `<conio.h>` digunakan agar menyediakan fungsi-fungsi yang berguna ketika ada interaksi langsung dengan keyboard, tanpa perlu menekan Enter. Terus `<stdlib.h>` digunakan untuk fungsi-fungsi manajemen memori, konversi angka, kontrol proses, dan lingkungan program.

Pada baris [5] terdapat `#define` yang mana digunakan untuk membuat sebuah konstanta, `MAX 20` yang ada menjadi penjelasan kalau kapasitas dari stack maksimal adalah 20 elemen.

Pada baris [7] terdapat `using namespace std;` yang mana digunakan untuk menghindari penulisan `std`

Pada baris [9] sampai [13] terdapat `struct stack` yang mana digunakan untuk menyimpan data dan info dari stack yang ada, dimana `int Atas` berguna untuk menyimpan posisi dari elemen paling atas pada stack dan `int Data[MAX]` berguna untuk menyimpan elemen pada stack sesuai dengan besar array yang ada.

Pada baris [15] terdapat `Stack Tumpuk` yang mana digunakan untuk membuat variabel yang bernama Tumpuk dari tipe data stack.

Pada baris [17] sampai [27] terdapat `int Kosong()` yang mana digunakan sebagai fungsi untuk mengecek apakah stack yang ada kosong atau tidak. Fungsi ini akan mengembalikan nilai 1 apabila nilai dari `Tumpuk. Atas` sama dengan -1, namun apabila nilai yang ada tidak sama dengan -1 fungsi akan mengembalikan nilai 0.

Pada baris [29] sampai [39] terdapat `int Penuh()` yang mana digunakan sebagai fungsi untuk mengecek apakah stack yang ada sudah penuh atau belum. Fungsi ini akan mengembalikan nilai 1 apabila nilai dari `Tumpuk. Atas` sama dengan `MAX - 1` atau 19, namun apabila nilai yang ada tidak sama dengan `MAX - 1` atau 19 fungsi akan mengembalikan 0.

Pada baris [41] sampai [59] terdapat `void Push()` yang mana digunakan sebagai fungsi untuk menambahkan nilai yang diinginkan ke dalam stack. Hal pertama yang dilakukan fungsi ini adalah mengecek apakah stack yang ada kosong, apabila stack yang ada kosong maka nilai yang ingin ditambahkan akan langsung

dimasukkan ke dalam stack. Kemudian, apabila di dalam stack masih ada ruang untuk menambahkan nilai, maka nilai yang ingin ditambahkan akan langsung dimasukkan lagi ke dalam stack. Terus apabila nilai tidak bisa untuk ditambahkan lagi, menandakan kalau stack yang ada penuh dan akan muncul pesan “stack telah penuh!!!”.

Pada baris [61] sampai [72] terdapat `void Pop()` yang mana digunakan sebagai fungsi untuk menghapus data yang ada di paling atas dari stack. Hal pertama yang dilakukan fungsi ini adalah mengecek apakah stack yang ada kosong, apabila stack yang ada tidak kosong maka nilai yang ada di tumpukan paling atas akan dihapus. Kemudian apabila stack yang ada dalam keadaan kosong, akan muncul pesan “Stack Kosong!!!”.

Pada baris [74] sampai [88] terdapat `void Cetak_Stack()` yang mana digunakan sebagai fungsi untuk menampilkan isi dari stack mulai dari atas ke bawah. Hal pertama yang dilakukan fungsi ini adalah mengecek apakah stack yang ada kosong, apabila stack yang ada tidak kosong maka `loop for` yang ada di dalam fungsi akan melakukan perulangan untuk menampilkan semua nilai yang telah diinput sebelumnya atau yang telah mengalami penghapusan mulai dari atas ke bawah. Kemudian apabila stack yang ada dalam keadaan kosong, akan muncul pesan “Stack Kosong!!!”.

Pada baris [90] sampai [94] terdapat `void Bersihkan_Stack()` yang mana digunakan sebagai fungsi untuk mengosongkan atau membersihkan semua nilai yang telah diinput ke dalam stack. `Tumpuk.Atas = -1` pada fungsi mengatur nilai indeks yang ada menjadi -1 yang mana tidak ada nilai pada indeks tersebut, apabila di isi kembali nilai yang ada akan dimasukkan ke dalam indeks 0 yang menjadi stack atas.

Pada baris [96] sampai [99] terdapat `void Inisialisasi()` yang mana digunakan sebagai fungsi untuk mengatur nilai awal dari stack agar berada di dalam keadaan kosong. Hal ini dilakukan dengan menetapkan nilai `Tumpuk.Atas = -1`. Nilai -1 menandakan bahwa tidak ada nilai yang tersimpan di dalam stack.

Pada baris [101] sampai [143] terdapat `int main()` yang mana digunakan untuk menjalankan dan menampilkan menu CLI. `Fungsi inisialisasi()` dipanggil di awal untuk mengatur stack agar berada di dalam kondisi kosong,

kemudian ada beberapa pilihan seperti *Push*, *Pop*, *Cetak*, *Bersihkan*, dan *Quit* yang dapat dipilih sesuai dengan *switch-case* yang diinput user. Terdapat *getch()* untuk menunggu tombol yang ditekan oleh pengguna dan membersihkan layar menggunakan *system("cls")*. Terus program akan terus berjalan selama user tidak memilih pilihan lima (5) untuk keluar atau menghentikan program yang ada.

SOAL 3

Buatlah program dengan tampilan sebagai berikut:

```
#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 15. Soal 3 Modul 2

A. Source Code

Tabel 5. Source Code Soal 3 Modul 2

```
1 #include <iostream>
2 #include <conio.h>
3 #include <stdlib.h>
4
5 #define MAX 20
6
7 using namespace std;
8
9 struct Queue
10 {
11     int Front, Rear, Size;
12     char Q[MAX];
13 };
14
15 Queue Antrian;
16
17 int Kosong()
18 {
19     if (Antrian.Front == Antrian.Rear)
20     {
21         return 1;
22     }
23     else
24     {
25         return 0;
26     }
27 }
28
29 int Penuh()
30 {
```

```

31     if  ((Antrian.Rear + 1) % Antrian.Size == 
32         Antrian.Front)
33     {
34         return 1;
35     }
36     else{
37         return 0;
38     }
39
40 void INSERT(char huruf)
41 {
42     if (Penuh() == 1)
43     {
44         cout << "Queue Penuh!!!" << endl;
45     }
46     else
47     {
48         Antrian.Q[Antrian.Rear] = huruf;
49         cout << "Data: " <<
50             Antrian.Q[Antrian.Rear] << " masuk ke dalam Queue"
51             << endl;
52         Antrian.Rear = (Antrian.Rear + 1) % 
53             Antrian.Size;
54     }
55 }
56 void DELETE()
57 {
58     if (Kosong() == 1)
59     {
60         cout << "Queue kosong!!!" << endl;

```

```

59     }
60     else
61     {
62         cout << "Data yang dihapus: " <<
63         Antrian.Q[Antrian.Front] << endl;
64         Antrian.Front = (Antrian.Front + 1) %
65         Antrian.Size;
66     }
67 }
68 void CETAKLAYAR()
69 {
70     if(Kosong() == 1)
71     {
72         cout << "Queue kosong!!!" << endl;
73     }
74     else
75     {
76         int i = Antrian.Front;
77         while(i != Antrian.Rear)
78         {
79             cout << "Queue ke-" << i << " = " <<
80             Antrian.Q[i] << endl;
81             i = (i + 1) % Antrian.Size;
82         }
83     }
84 void RESET()
85 {
86     Antrian.Front = 0;
87     Antrian.Rear = 0;

```

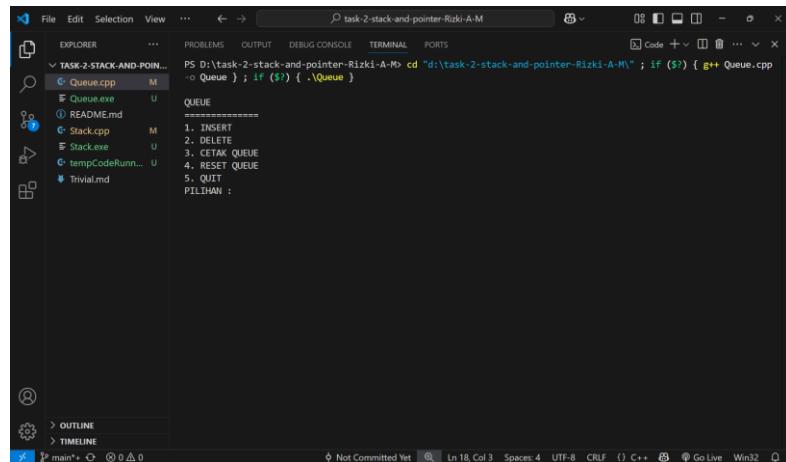
```

88     Antrian.Size = MAX;
89     cout << "Queue telah di-reset" << endl;
90 }
91
92 void Inisialisasi()
93 {
94     Antrian.Front = 0;
95     Antrian.Rear = 0;
96     Antrian.Size = MAX;
97 }
98
99 int main()
100 {
101     Inisialisasi();
102     int Pilihan;
103     char huruf;
104     do{
105         cout << "\nQUEUE" << endl;
106         cout << "======" << endl;
107         cout << "1. INSERT" << endl;
108         cout << "2. DELETE" << endl;
109         cout << "3. CETAK QUEUE" << endl;
110         cout << "4. RESET QUEUE" << endl;
111         cout << "5. QUIT" << endl;
112         cout << "PILIHAN : "; cin >> Pilihan;
113         switch (Pilihan)
114         {
115             case 1:
116                 cout << "Masukkan Nilai: "; cin >>
huruf;
117                 INSERT(huruf);
118                 break;

```

```
119
120     case 2:
121         DELETE();
122         break;
123
124     case 3:
125         CETAKLAYAR();
126         break;
127
128     case 4:
129         RESET();
130         break;
131
132     default:
133         cout << "TERIMA KASIH" << endl;
134         break;
135     }
136     cout << "Press any key to continue";
137     getch();
138     system("cls");
139 }
140 while (Pilihan < 5);
141 return 0;
142 }
```

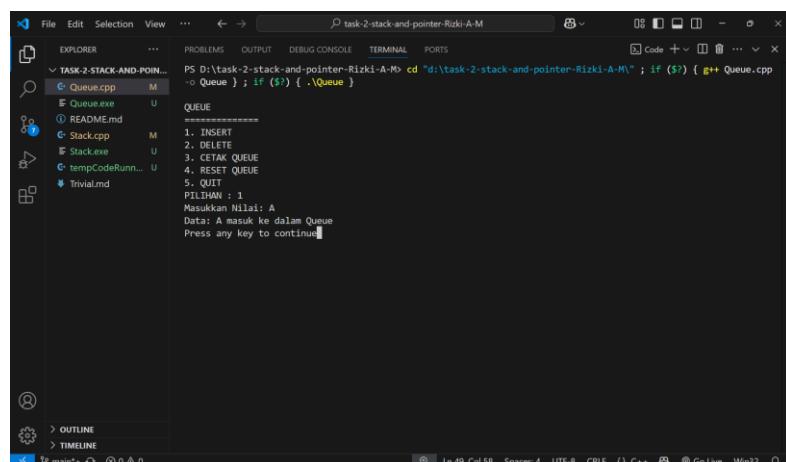
B. Output Program



```
File Edit Selection View ... ← → ⌂ task-2-stack-and-pointer-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\task-2-stack-and-pointer-Rizki-A-M> cd "d:\task-2-stack-and-pointer-Rizki-A-M\"; IF ($?) { g++ Queue.cpp
-o Queue } ; IF ($?) { .\Queue }

QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN :
```

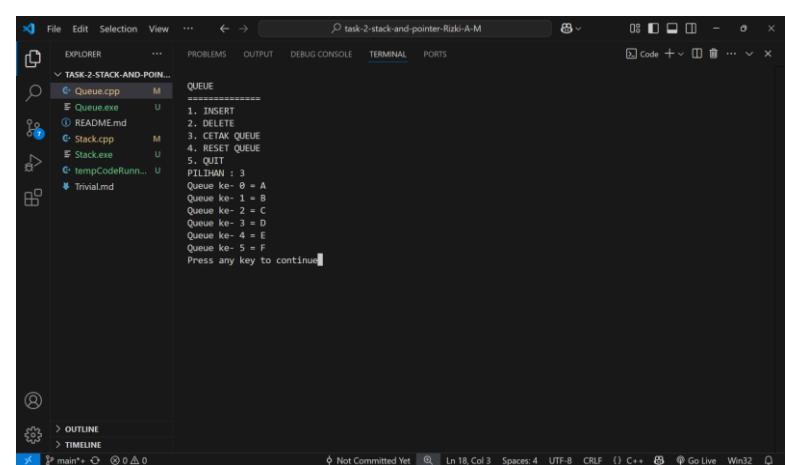
Gambar 16. Tampilan Awal Program Saat Dijalankan



```
File Edit Selection View ... ← → ⌂ task-2-stack-and-pointer-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\task-2-stack-and-pointer-Rizki-A-M> cd "d:\task-2-stack-and-pointer-Rizki-A-M\"; IF ($?) { g++ Queue.cpp
-o Queue } ; IF ($?) { .\Queue }

QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 1
Masukkan Nilai: A
Data: A masuk ke dalam Queue
Press any key to continue
```

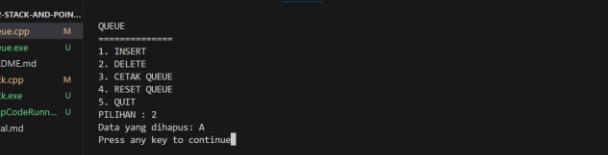
Gambar 17. Memasukkan Huruf ke Dalam Queue



```
File Edit Selection View ... ← → ⌂ task-2-stack-and-pointer-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\task-2-stack-and-pointer-Rizki-A-M> cd "d:\task-2-stack-and-pointer-Rizki-A-M\"; IF ($?) { g++ Queue.cpp
-o Queue } ; IF ($?) { .\Queue }

QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 3
Queue Ke- 0 = A
Queue Ke- 1 = B
Queue Ke- 2 = C
Queue Ke- 3 = D
Queue Ke- 4 = E
Queue Ke- 5 = F
Press any key to continue
```

Gambar 18. Menampilkan Queue Yang Sudah Dimasukkan



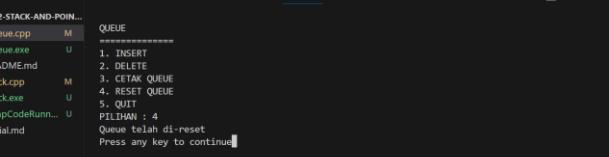
The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure for "TASK-2-STACK-AND-POIN...". The "Queue.cpp" file is the active file, indicated by a green dot in the icon. Other files shown include "Queue.exe", "README.md", "Stack.cpp", "Stack.exe", "tempCodeRunn...", and "Trivial.md".
- Terminal:** The title bar says "task-2-stack-and-pointer-Rizki-A-M". The terminal content is as follows:

```
QUEUE
-----
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 2
Data yang dihapus: A
Press any key to continue
```
- Bottom Status Bar:** Shows "Not Committed Yet", "Ln 18, Col 3", "Spaces: 4", "UTF-8", "CRLF", "C++", "Go Live", and "Win32".

Gambar 19. Melakukan Delete Pada Huruf yang Sudah Dimasukkan

Gambar 20. Tampilan Queue Setelah Melakukan Delete



task-2-stack-and-pointer-Rizki-A-M

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Code +

EXPLORER

TASK-2-STACK-AND-POIN...

Queue.cpp M

Queue.exe U

README.md

Stack.cpp M

Stack.exe U

tempCodeRunn... U

Trivial.md

QUEUE

=====

1. INSERT

2. DELETE

3. CETAK QUEUE

4. RESET QUEUE

5. QUIT

PILIHAN : 4

Queue telah di-reset

Press any key to continue

main.cpp

Ln 18, Col 3

Spaces: 4

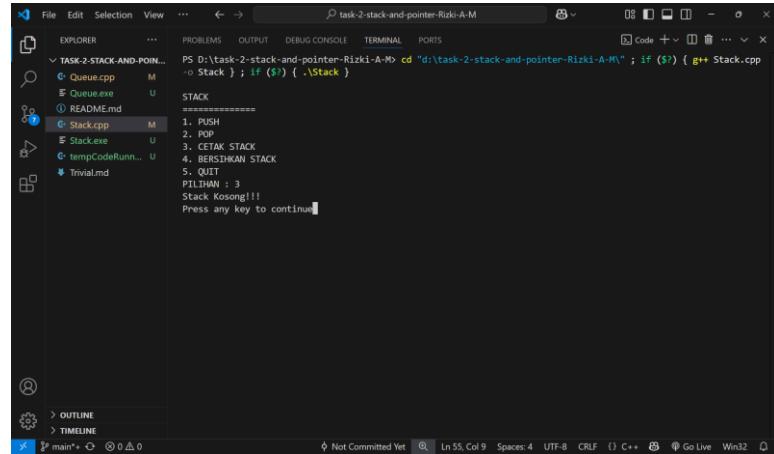
UTF-8

CRLF

Go Live

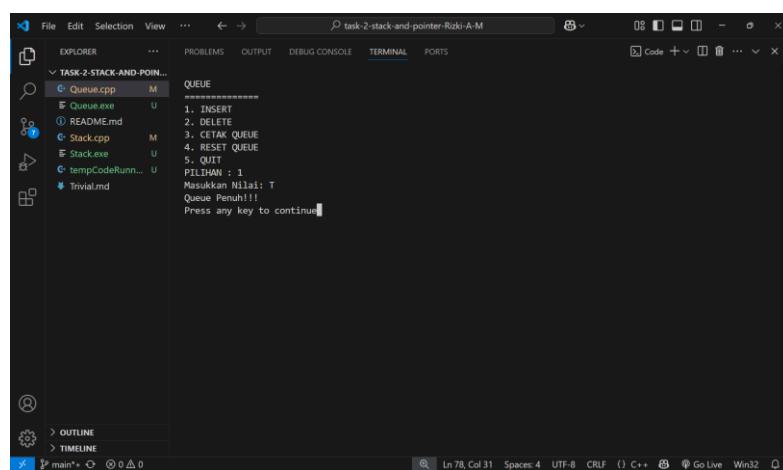
Win32

Gambar 21. Mereset Queue



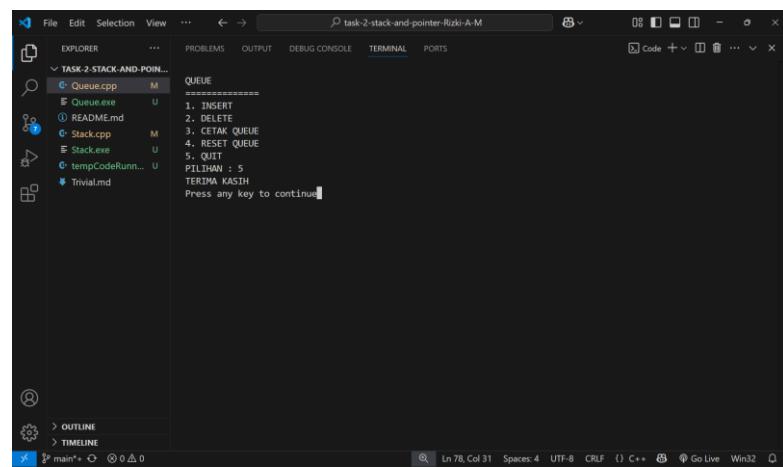
```
PS D:\task-2-stack-and-pointer-Rizki-A-M> cd "d:\task-2-stack-and-pointer-Rizki-A-M\"; if ($?) { g++ Stack.cpp
Stack
=====
1. PUSH
2. POP
3. CETAK STACK
4. BERSIHKAN STACK
5. QUIT
PILIHAN : 3
Stack Kosong!!!
Press any key to continue
```

Gambar 22. Tampilan Queue Setelah di Reset



```
PS D:\task-2-stack-and-pointer-Rizki-A-M> cd "d:\task-2-stack-and-pointer-Rizki-A-M\"; if ($?) { g++ Queue.cpp
QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 1
Masukan Nilai: T
Queue Penuh!!!
Press any key to continue
```

Gambar 23. Tampilan Jika Queue Penuh



```
PS D:\task-2-stack-and-pointer-Rizki-A-M> cd "d:\task-2-stack-and-pointer-Rizki-A-M\"; if ($?) { g++ Queue.cpp
QUEUE
=====
1. INSERT
2. DELETE
3. CETAK QUEUE
4. RESET QUEUE
5. QUIT
PILIHAN : 5
TERIMA KASIH
Press any key to continue
```

Gambar 24. Tampilan Saat Memilih Menu 5 Quit

C. Pembahasan

Pada baris [1] sampai [3] terdapat `#include` yang mana digunakan untuk mengakses sebuah file yang diinginkan. `<iostream>` yang ada digunakan untuk input dan output. Kemudian `<conio.h>` digunakan agar menyediakan fungsi-fungsi yang berguna ketika ada interaksi langsung dengan keyboard, tanpa perlu menekan Enter. Terus `<stdlib.h>` digunakan untuk fungsi-fungsi manajemen memori, konversi angka, kontrol proses, dan lingkungan program.

Pada baris [5] terdapat `#define` yang mana digunakan untuk membuat sebuah konstanta, `MAX 20` yang ada menjadi penjelasan kalau kapasitas dari stack maksimal adalah 20 elemen.

Pada baris [7] terdapat `using namespace std;` yang mana digunakan untuk menghindari penulisan `std`

Pada baris [9] sampai [13] terdapat `struct Queue` yang mana digunakan untuk menyimpan data dan info antrian, dimana `Front` berguna untuk menunjukkan posisi data terdepan di dalam queue. Kemudian `Rear` berguna untuk menunjukkan posisi data terakhir di dalam queue, `Size` berguna untuk menerangkan seberapa banyak data yang dapat ditampung pada queue, dan `Q[MAX]` berguna untuk menyimpan elemen pada stack sesuai dengan besar array yang ada.

Pada baris [15] terdapat `Queue Antrian` yang mana digunakan untuk membuat variabel yang bernama Tumpuk dari tipe data stack.

Pada baris [17] sampai [27] terdapat `int Kosong()` yang mana digunakan sebagai fungsi untuk mengecek apakah queue yang ada kosong atau tidak. Fungsi ini akan mengembalikan nilai 1 apabila nilai dari `antrean.Front` sama dengan `antrean.Rear`. Begitu juga sebaliknya apabila nilai yang ada tidak sama dengan, fungsi akan mengembalikan nilai 0.

Pada baris [29] sampai [38] terdapat `int Penuh()` yang mana digunakan sebagai fungsi untuk mengecek apakah queue yang ada sudah penuh atau belum. Fungsi ini akan mengembalikan nilai 1 apabila kondisi dari `(Antrian.Rear + 1) % Antrian.Size == Antrian.Front` terpenuhi, apabila tidak terpenuhi maka fungsi akan mengembalikan nilai 0.

Pada baris [40] sampai [52] terdapat `void INSERT(char huruf)` yang mana digunakan sebagai fungsi untuk menambahkan huruf yang diinginkan ke dalam queue. Hal pertama yang dilakukan fungsi ini adalah mengecek apakah queue yang ada kosong, apabila queue yang ada kosong maka huruf yang ingin ditambahkan ke dalam queue. Terus apabila huruf yang diinginkan tidak bisa lagi untuk ditambahkan, itu menandakan kalau queue yang ada penuh dan akan muncul pesan “Queue Penuh!!!”.

Pada baris [54] sampai [65] terdapat `void DELETE()` yang mana digunakan sebagai fungsi untuk menghapus data yang posisinya berada di paling depan pada queue. Hal pertama yang dilakukan fungsi ini adalah mengecek apakah queue yang ada kosong, apabila queue yang ada tidak kosong maka huruf yang ada di paling depan akan dihapus. Kemudian apabila queue yang ada dalam keadaan kosong, akan muncul pesan “Queue kosong!!!”.

Pada baris [67] sampai [82] terdapat `void CETAKLAYAR()` yang mana digunakan sebagai fungsi untuk menampilkan isi dari queue mulai dari yang pertama dimasukkan sampai yang terakhir dimasukkan (sesuai dengan urutan diinput). Hal pertama yang dilakukan fungsi ini adalah mengecek apakah queue yang ada kosong, apabila queue yang ada tidak kosong maka `loop` yang ada akan melakukan perulangan dalam menampilkan semua nilai yang telah diinput sebelumnya atau yang telah mengalami penghapusan mulai dari atas ke bawah. Kemudian apabila queue yang ada dalam keadaan kosong, akan muncul pesan “Queue kosong!!!”.

Pada baris [84] sampai [90] terdapat `void RESET()` yang mana digunakan sebagai fungsi untuk mengosongkan atau membersihkan semua yang telah diinput ke dalam queue. `antrean.Front = 0` dan `antrean.Rear = 0` pada fungsi untuk mengatur ulang posisi `Front` dan `Rear` kembali ke indeks 0, yang mana sama seperti kondisi awal sebelum diinput.

Pada baris [92] sampai [97] terdapat `void Inisialisasi()` yang mana digunakan sebagai fungsi untuk mengatur ulang posisi `Front` dan `Rear` kembali ke indeks 0.

Pada baris [99] sampai [142] terdapat `int main()` yang mana digunakan untuk menjalankan dan menampilkan menu CLI. `Fungsi inisialisasi()`

dipanggil di awal untuk mengatur queue agar berada di dalam kondisi kosong, kemudian ada beberapa pilihan seperti *Insert*, *Delete*, *Cetaklayar*, *Reset*, dan *Quit* yang dapat dipilih sesuai dengan *switch-case* yang diinput user. Pada bagian *switch-case* user diminta untuk memasukkan sebuah string dan hanya karakter pertama yang di masukkan pada program yang akan diproses untuk dimasukkan ke dalam queue. Kemudian terdapat *getch()* untuk menunggu tombol yang ditekan oleh pengguna dan membersihkan layar menggunakan *system("cls")*. Terus program akan terus berjalan selama user tidak memilih pilihan lima (5) untuk keluar atau menghentikan program yang ada.

MODUL 3 : SINGLE LINK LIST

SOAL 1

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

```
1 #include <iostream.h>
2 #include <iostream>
3 #include <stdlib.h>
4
5 using namespace std;
6
7 typedef struct Node {
8     string data;
9     Node *next;
10 } ;
11
12 Node *head, *tail;
13
14 int pilih;
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 cariData();
27 void hapusData();
28 void sisipkanSebelum();
29 void sisipkanSetelah();
30
31 int main()
32 {
33     do {
34         cout<<"Single Linked List Circular (SLIC)"<<endl;
35         cout<<".>> 1. Tambah Depan"<<endl;
36         cout<<".>> 2. Tambah Belakang"<<endl;
37         cout<<".>> 3. Hapus Depan"<<endl;
38         cout<<".>> 4. Hapus Belakang"<<endl;
39         cout<<".>> 5. Tampilkan Data"<<endl;
40         cout<<".>> 6. Hapus Semua Elemen"<<endl;
41         cout<<".>> 7. Cari Data"<<endl;
42         cout<<".>> 8. Sisipkan Data Baru Sebelum Data Tertentu"<<endl;
43         cout<<".>> 9. Sisipkan Node/Data Baru Sebelum Data Tertentu"<<endl;
44         cout<<".>> 10. Sisipkan Node/Data Baru Setelah Data Tertentu"<<endl;
45         cout<<".>> 11. Hapus . . .>> 12. Quit"<<endl;
46         cout<<".>> 13. Tambah . . .>> 14. Hapus . . .>> 15. Cari Data . . .>> 16. Sisipkan Data . . .>> 17. Hapus . . .>> 18. Tampilkan . . .>> 19. Hapus Semua Elemen . . .>> 20. Cari Data . . .>> 21. Sisipkan Data . . .>> 22. Hapus . . .>> 23. Tampilkan . . .>> 24. Hapus . . .>> 25. Cari Data . . .>> 26. Sisipkan Data . . .>> 27. Hapus . . .>> 28. Tampilkan . . .>> 29. Hapus . . .>> 30. Cari Data . . .>> 31. Sisipkan Data . . .>> 32. Hapus . . .>> 33. Tampilkan . . .>> 34. Hapus . . .>> 35. Cari Data . . .>> 36. Sisipkan Data . . .>> 37. Hapus . . .>> 38. Tampilkan . . .>> 39. Hapus . . .>> 40. Cari Data . . .>> 41. Sisipkan Data . . .>> 42. Hapus . . .>> 43. Tampilkan . . .>> 44. Hapus . . .>> 45. Cari Data . . .>> 46. Sisipkan Data . . .>> 47. Hapus . . .>> 48. Tampilkan . . .>> 49. Hapus . . .>> 50. Cari Data . . .>> 51. Sisipkan Data . . .>> 52. Hapus . . .>> 53. Tampilkan . . .>> 54. Hapus . . .>> 55. Cari Data . . .>> 56. Sisipkan Data . . .>> 57. Hapus . . .>> 58. Tampilkan . . .>> 59. Hapus . . .>> 60. Cari Data . . .>> 61. Sisipkan Data . . .>> 62. Hapus . . .>> 63. Tampilkan . . .>> 64. Hapus . . .>> 65. Cari Data . . .>> 66. Sisipkan Data . . .>> 67. Hapus . . .>> 68. Tampilkan . . .>> 69. Hapus . . .>> 70. Cari Data . . .>> 71. Sisipkan Data . . .>> 72. Hapus . . .>> 73. Cari Data . . .>> 74. Sisipkan Data . . .>> 75. Hapus . . .>> 76. Tampilkan . . .>> 77. Hapus . . .>> 78. Cari Data . . .>> 79. Sisipkan Data . . .>> 80. Hapus . . .>> 81. Tampilkan . . .>> 82. Hapus . . .>> 83. Cari Data . . .>> 84. Sisipkan Data . . .>> 85. Hapus . . .>> 86. Tampilkan . . .>> 87. Hapus . . .>> 88. Cari Data . . .>> 89. Sisipkan Data . . .>> 90. Hapus . . .>> 91. Cari Data . . .>> 92. Sisipkan Data . . .>> 93. Hapus . . .>> 94. } while (pilih != 12);
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     cin>>dataBaru;
110     baru = new Node;
111     baru->data = dataBaru;
112     baru->next = baru;
113     baru->next = baru;
114
115     if((isEmpty() == 1) {
116         head = baru;
117         tail = baru;
118     } else {
119         head->next = baru;
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     Node *baru;
129     baru = new Node;
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     }
141 }
```

```

141     } else {
142         cout << "Data " << dataBaris << " berhasil dimasukkan di bagian belakang." << endl;
143     }
144 }
145 void hapusDepan() {
146     if((isEmpty()) == 0) {
147         Node *hapus;
148         hapus = head;
149         data = hapus->data;
150         tail = hapus->next;
151         if(head != tail) {
152             head = head->next;
153             tail->next = head;
154         } else {
155             init();
156         }
157         delete hapus;
158     } else cout << "Tidak terdapat data pada Linked List." << endl;
159 }
160 void hapusBelakang() {
161     if((isEmpty()) == 0) {
162         Node *hapus, *newtail;
163         hapus = head;
164         data = hapus->data;
165         if(head == tail) {
166             newtail = head;
167             while(newtail->next != tail) {
168                 newtail = newtail->next;
169             }
170             tail = newtail;
171             tail->next = head;
172         } else {
173             init();
174         }
175         delete hapus;
176     } else cout << "Tidak terdapat data pada Linked List." << endl;
177 }
178 void tambahkan() {
179     if((isEmpty()) == 0) {
180         Node *bantu;
181
182         bantu = head;
183
184         do {
185             cout << "Data yang ingin ditambahkan : ";
186             cin >> data;
187             bantu = bantu->next;
188         } while(bantu != head);
189         cout << endl;
190     } else cout << "Tidak terdapat data pada Linked List." << endl;
191 }
192 void reset() {
193     if((isEmpty()) == 0) {
194         Node *bantu, *hapus;
195         bantu = head;
196
197         do {
198             hapus = bantu;
199             bantu = bantu->next;
200             delete hapus;
201         } while(bantu != head);
202
203         init();
204         cout << "Seluruh elemen pada Linked List telah dibersihkan." << endl;
205     } else cout << "Tidak terdapat data pada Linked List." << endl;
206 }
207 void cariData() {
208     if((isEmpty()) == 0) {
209         string cari;
210         cout << "Masukkan data yang ingin dicari : ";
211         cin >> cari;
212
213         Node *bantu, *hapus, *newtail, *bantuTampilkan;
214         bool apabilaTemukan = false;
215
216         bantu = head;
217
218         do {
219             if(cari == bantu->data) {
220                 cout << "Data yang dicari berhasil di dalam linked list berada di index " << i << "..." <<
221                 << " merupakan data yang anda cari." << endl;
222                 cout << "Linked List : ";
223                 bantuTampilkan = head;
224
225                 do {
226                     if(cari == bantuTampilkan->data)
227                         cout << "[<<bantuTampilkan->data<>]" << " ";
228                     else
229                         cout << bantuTampilkan->data << " ";
230                     bantuTampilkan = bantuTampilkan->next;
231                 } while(bantuTampilkan != head);
232
233                 apabilaTemukan = true;
234                 cout << endl;
235                 break;
236             }
237             bantu = bantu->next;
238         } while(bantu != head);
239
240         if(apabilaTemukan == false)
241             cout << "Data " << cari << " tidak ditemukan pada Linked List." << endl;
242     } else cout << "Tidak terdapat data pada Linked List." << endl;
243 }
244
245 void hapusData() {
246     if((isEmpty()) == 0) {
247         string cari;
248         cout << "Masukkan data yang ingin dihapus : ";
249         cin >> cari;
250
251         Node *bantu, *sebelum, *hapus[255], *bantu2;
252         int hitung = 0;
253         bool apabilaTemukan = false;
254
255         bantu = head;
256
257         do {
258             bantu2 = bantu;
259             if((car1 == bantu->data) {
260                 hapus[hitung] = bantu;
261                 apabilaTemukan = true;
262                 if((bantu != head && bantu != tail) {
263                     sebelum->next = bantu->next;
264                     bantu2 = sebelum;
265                 }
266             }
267             sebelum = bantu2;
268             bantu = bantu->next;
269         } while(bantu != head);
270
271         if(apabilaTemukan == true) {
272             if((hitung < 0) || (hitung > 255)) {
273                 cout << "Data " << cari << " berhasil dihapus." << endl;
274             }
275         }
276     }
277 }

```

```

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

Gambar 25 Soal 1 Modul 3

A. Source Code

Tabel 6. Source Code Soal 1 Modul 3

```
1 #include <conio.h>
2 #include <iostream>
3 #include <stdlib.h>
4
5 using namespace std;
6
7 typedef struct TNode
```

```

8  {
9      string data;
10     TNode *next;
11  };
12
13 TNode *head, *tail;
14
15 int pil;
16 char pilihan [2];
17 string dataBaru, dataDelete;
18
19 void Init();
20 int Kosong();
21
22 void TambahDepan();
23 void TambahBelakang();
24 void HapusDepan();
25 void HapusBelakang();
26 void Tampilkan();
27 void Reset();
28 void CariData();
29 void HapusData();
30 void SisipkanSebelum();
31 void SisipkanSetelah();
32
33 int main()
34 {
35     do
36     {
37         cout<<"Single     Linked     List     Circular
(SLLC)"<<endl;

```

```

38     cout<<"======"<<endl;
39         cout<<"1. Tambah Depan"<<endl;
40         cout<<"2. Tambah Belakang"<<endl;
41         cout<<"3. Hapus Depan"<<endl;
42         cout<<"4. Hapus Belakang"<<endl;
43         cout<<"5. Tampilkan Data"<<endl;
44         cout<<"6. Hapus Semua elemen"<<endl;
45         cout<<"7. Cari Data"<<endl;
46         cout<<"8.      Hapus      Setiap      Data
Tertentu"<<endl;
47         cout<<"9. Sisipkan Node/Data Baru Sebelum
Data Tertentu"<<endl;
48         cout<<"10. Sisipkan Node/Data Baru Setelah
Data Tertentu"<<endl;
49         cout<<"11. Keluar"<<endl;
50
51     cout<<"======"<<endl;
52         cout<<"Pilihan : ";
53         cin>>pilihan;
54         pil = atoi(pilihan);
55
56         switch(pil)
57         {
58             case 1:
59                 TambahDepan();
60                 break;
61             case 2:
62                 TambahBelakang();
63                 break;
64             case 3:
65                 HapusDepan();

```

```

65           cout<<"Data \"<<dataDelete<<"\" yang
berada di depan telah berhasil dihapus."<<endl;
66           break;
67           case 4:
68               HapusBelakang();
69               cout<<"Data \"<<dataDelete<<"\" yang
berada di belakang telah berhasil dihapus."<<endl;
70           break;
71           case 5:
72               Tampilkan();
73           break;
74           case 6:
75               Reset();
76           break;
77           case 7:
78               CariData();
79           break;
80           case 8:
81               HapusData();
82           break;
83           case 9:
84               SisipkanSebelum();
85           break;
86           case 10:
87               SisipkanSetelah();
88           break;
89           default:
90               cout<<"\nTERIMA KASIH"<<endl;
91               cout<<"Program was made by Rizki
Adhitiya Maulana (2410817110014)"<<endl;
92           }
93

```

```

94         cout<<"\nPress      any      key      to
95         continue!"<<endl;
96         getch();
97         system("cls");
98         }
99     }
100
101 void Init()
102 {
103     head = NULL;
104     tail = NULL;
105 }
106
107 int Kosong()
108 {
109     if(head == NULL)
110         return 1;
111     else
112         return 0;
113 }
114
115 void TambahDepan()
116 {
117     cout<<"Masukkan Data : ";
118     TNode *baru;
119     baru = new TNode;
120     cin>>dataBaru;
121     baru->data = dataBaru;
122     baru->next = baru;
123
124     if(Kosong() == 1)

```

```

125     {
126         head = baru;
127         tail = baru;
128     }
129     else
130     {
131         baru->next = head;
132         head = baru;
133         tail->next = head;
134     }
135     cout<<"Data \"<<dataBaru<<\" telah berhasil
136     dimasukkan di bagian depan."<<endl;
137 }
138 void TambahBelakang()
139 {
140     cout<<"Masukkan Data : ";
141     TNode *baru;
142     baru = new TNode;
143     cin>>dataBaru;
144     baru->data = dataBaru;
145     baru->next = baru;
146
147     if(Kosong() == 1)
148     {
149         head = baru;
150         tail = baru;
151     }
152     else
153     {
154         tail->next = baru;
155         tail = baru;

```

```

156         tail->next = head;
157     }
158     cout<<"Data \"<<dataBaru<<\" telah berhasil
159     dimasukkan di bagian belakang."<<endl;
160 }
161 void HapusDepan()
162 {
163     if(Kosong() == 0)
164     {
165         TNode *hapus;
166         hapus = head;
167         dataDelete = hapus->data;
168
169         if(head != tail)
170         {
171             head = head->next;
172             tail->next = head;
173         }
174     else
175     {
176         Init();
177     }
178
179     delete hapus;
180 }
181 else
182 {
183     cout<<"Tidak terdapat data pada Linked
184     List."<<endl;
185 }

```

```

186
187 void HapusBelakang()
188 {
189     if(Kosong() == 0)
190     {
191         TNode *hapus, *newTail;
192         hapus = tail;
193         dataDelete = hapus->data;
194
195         if(head != tail)
196         {
197             newTail = head;
198             while(newTail->next != tail)
199             {
200                 newTail = newTail->next;
201             }
202             tail = newTail;
203             tail->next = head;
204         }
205         else
206         {
207             Init();
208         }
209         delete hapus;
210     }
211     else
212     {
213         cout<<"Tidak terdapat data pada Linked
214             List."<<endl;
215     }
216

```

```

217 void Tampilan()
218 {
219     if(Kosong() == 0)
220     {
221         TNode *bantu;
222         bantu = head;
223
224         do
225         {
226             cout<<bantu->data<< ' ';
227             bantu = bantu->next;
228         }
229         while (bantu != head);
230         cout<<endl;
231     }
232     else
233     {
234         cout<<"Tidak terdapat data pada Linked
235 List."<<endl;
236     }
237
238 void Reset()
239 {
240     if(Kosong() == 0)
241     {
242         TNode *bantu, *hapus;
243         bantu = head;
244
245         do
246         {
247             hapus = bantu;

```

```

248         bantu = bantu->next;
249         delete hapus;
250     }
251     while (bantu != head);
252     Init();
253     cout<<"Seluruh elemen pada Linked List
254     telah dibersihkan."<<endl;
255 }
256 {
257     cout<<"Tidak terdapat data pada Linked
258     List."<<endl;
259 }
260
261 void CariData()
262 {
263     if(Kosong() == 0)
264     {
265         string cari;
266         cout<<"Masukkan data yang ingin dicari :
267         ";
268
269         TNode      *bantu,      *hapus,      *newTail,
270         *bantuTampilan;
271
272         bantu = head;
273
274         do
275         {

```

```

276         if(cari == bantu->data)
277         {
278             cout<<"Setiap data yang berada di
279             dalam tanda kurung siku ([...]) "
280             <<"merupakan data yang anda
281             cari."<<endl;
282
283             cout<<"Linked List : ";
284             bantuTampilan = head;
285
286             do
287             {
288                 if(cari == bantuTampilan-
289                 >data)
290                 {
291                     cout<<" [ "<<bantuTampilan->data<<"] ";
292                     }
293                     else
294                     {
295                         cout<<bantuTampilan-
296                         >data<<' ' ;
297                         }
298                     bantuTampilan
299                     =
300                     bantuTampilan->next;
301                     }
302                     while (bantuTampilan != head);
303
304                     apaDitemukan = true;
305                     cout<<endl;
306                     break;
307                 }
308             bantu = bantu->next;

```

```

302         }
303         while (bantu != head);
304
305         if(apaditemukan == false)
306         {
307             cout<<"Data     \\"<<cari<<"\"     tidak
308             ditemukan pada Linked List."<<endl;
309         }
310     else
311     {
312         cout<<"Tidak terdapat data pada Linked
313             List."<<endl;
314     }
315
316 void HapusData()
317 {
318     if(Kosong() == 0)
319     {
320         string cari;
321         cout<<"Masukkan data yang ingin dihapus :
322         ";
323
324         TNode *bantu, *sebelum, *hapus[255],
325         *bantu2;
326         int hitung = 0;
327         bool apaditemukan = false;
328
329         bantu = head;

```

```

330      do
331      {
332          bantu2 = bantu;
333          if(cari == bantu->data)
334          {
335              hapus[hitung++] = bantu;
336              apaDitemukan = true;
337              if(bantu != head && bantu != tail)
338              {
339                  sebelum->next = bantu->next;
340                  bantu2 = sebelum;
341              }
342          }
343          sebelum = bantu2;
344          bantu = bantu->next;
345      }
346      while (bantu != head);

347

348      if(apaditemukan == true)
349      {
350          for(int i=0; i<hitung; i++)
351          {
352              if(hapus[i] == head)
353              {
354                  HapusDepan();
355              }
356              else if(hapus[i] == tail)
357              {
358                  HapusBelakang();
359              }
360              else
361              {

```

```

362                     delete hapus[i];
363
364                 }
365                 cout<<"Setiap data \""<<cari<<"\
366                 yang terdapat pada Linked List telah
367                 dihapus."<<endl;
368             }
369             else
370             {
371                 cout<<"Data \""<<cari<<"\" tidak
372                 ditemukan pada Linked List."<<endl;
373             }
374             cout<<"Tidak terdapat data pada Linked
375             List."<<endl;
376         }
377
378 void SisipkanSebelum()
379 {
380     if(Kosong() == 0)
381     {
382         TNode *bantu, *sebelum;
383         string nextData;
384         bool apaAda;
385
386         bantu = head;
387         sebelum = tail;
388

```

```

389         cout<<"Sisipkan data baru sebelum data : "
400         ;
401         cin>>nextData;
402
403         do
404         {
405             if(nextData == bantu->data)
406             {
407                 apaAda = true;
408                 break;
409             }
410             else
411             {
412                 sebelum = bantu;
413                 bantu = bantu->next;
414             }
415         }
416         while (bantu != head);
417
418         if(apaAda == true)
419         {
420             cout<<"Masukkan data yang ingin
421 ditambahkan : ";
422             cin>>dataBaru;
423
424             TNode *baru;
425             baru = new TNode;
426
427             baru->data = dataBaru;
428             baru->next = bantu;
429
430             sebelum->next = baru;

```

```

419
420             if(bantu == head)
421             {
422                 head = baru;
423             }
424             cout<<"Data      \\""<<<dataBaru<<"\"
425             berhasil      disisipkan      sebelum      data
426             \""<<nextData<<"\."<<endl;
427         }
428     else
429     {
430         cout<<"Tidak      terdapat      data
431             \""<<nextData<<"\"
432             pada Linked List."<<endl;
433     }
434 }
435 }
436
437 void SisipkanSetelah()
438 {
439     if(Kosong() == 0)
440     {
441         TNode *bantu;
442         string prevData;
443         bool apaAda;
444
445         bantu = head;
446

```

```

447         cout<<"Sisipkan data baru setelah data : "
448         ";
449
450         do
451         {
452             if(prevData == bantu->data)
453             {
454                 apaAda = true;
455                 break;
456             }
457             else
458             {
459                 bantu = bantu->next;
460             }
461         }
462         while (bantu != head);
463
464         if(apaAda == true)
465         {
466             cout<<"Masukkan data yang ingin
467 ditambahkan : ";
468
469             TNode *baru;
470             baru = new TNode;
471
472             baru->data = dataBaru;
473             baru->next = bantu->next;
474
475             bantu->next = baru;
476

```

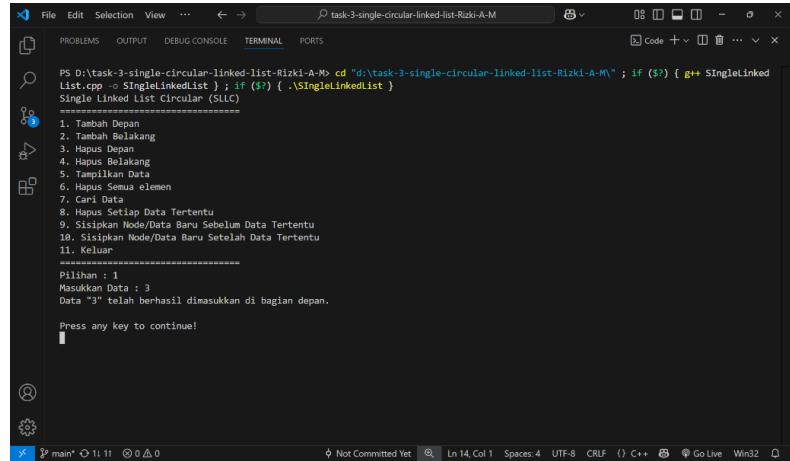
```

477         if(bantu == tail)
478         {
479             tail = baru;
480         }
481         cout<<"Data      ""<<dataBaru<<"\""
482         berhasil      disisipkan      sebelum      data
483         \""<<prevData<<"\"."<<endl;
484     }
485     else
486     {
487         cout<<"Tidak      terdapat      data
488         \""<<prevData<<"\" pada Linked List."<<endl;
489     }
490     else
491     {
492         cout<<"Tidak      terdapat      data pada Linked
493         List."<<endl;
494     }

```

B. Output Program

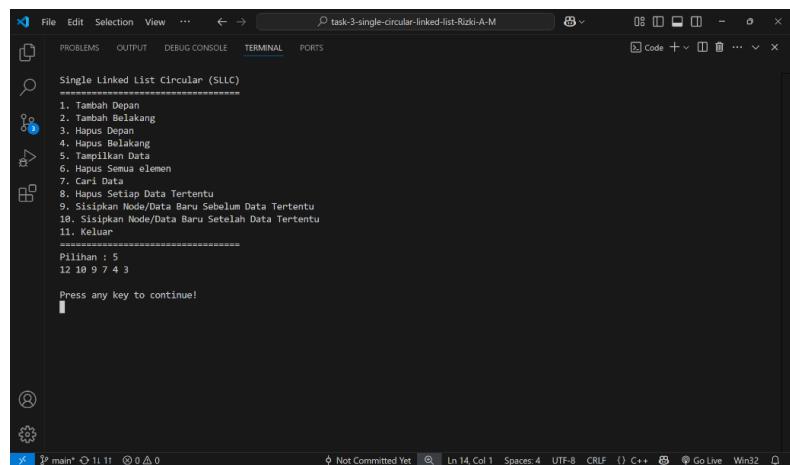
Gambar 26. Tampilan Awal Program



```
PS D:\task-3-single-circular-linked-list-Rizki-A-M> cd "d:\task-3-single-circular-linked-list-Rizki-A-M\" ; if ($?) { g++ SIngleLinkedList
List.cpp -o SingleLinkedList } ; if ($?) { .\SingleLinkedList
Single Linked List Circular (SLC)
=====
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. Keluar
=====
Pilihan : 1
Masukkan Data : 3
Data "3" telah berhasil dimasukkan di bagian depan.

Press any key to continue!
```

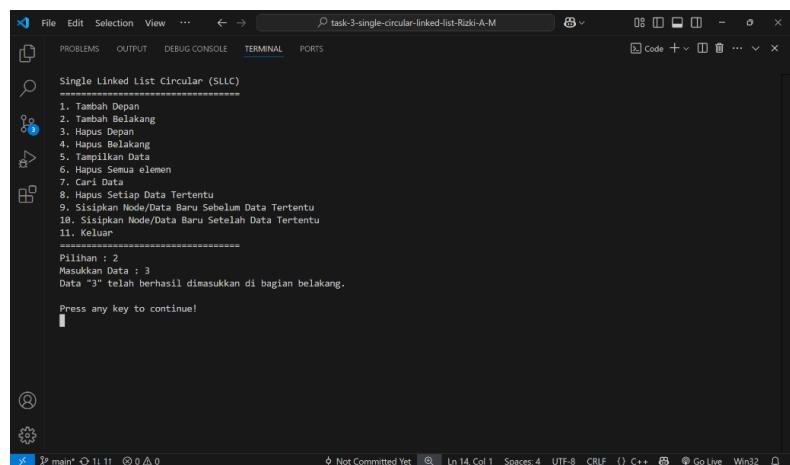
Gambar 27. Tampilan Tambah Data dari Depan



```
PS D:\task-3-single-circular-linked-list-Rizki-A-M> cd "d:\task-3-single-circular-linked-list-Rizki-A-M\" ; if ($?) { g++ SIngleLinkedList
List.cpp -o SingleLinkedList } ; if ($?) { .\SingleLinkedList
Single Linked List Circular (SLC)
=====
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. Keluar
=====
Pilihan : 5
12 10 9 7 4 3

Press any key to continue!
```

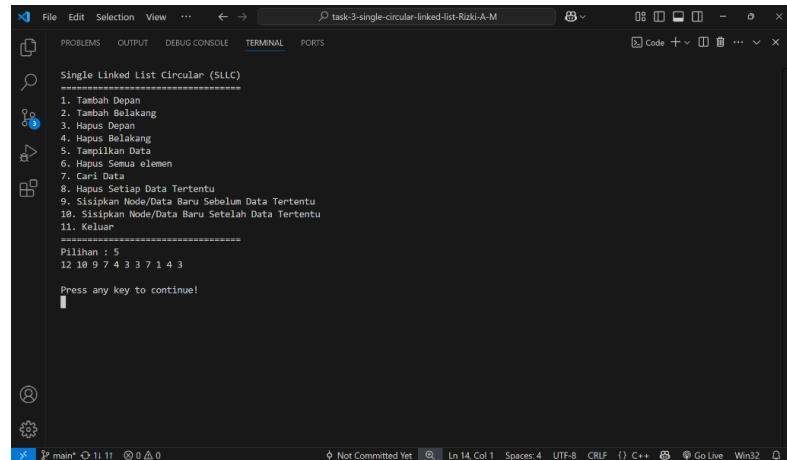
Gambar 28. Tampilan Data Setelah Dilakukan Tambah Data Depan



```
PS D:\task-3-single-circular-linked-list-Rizki-A-M> cd "d:\task-3-single-circular-linked-list-Rizki-A-M\" ; if ($?) { g++ SIngleLinkedList
List.cpp -o SingleLinkedList } ; if ($?) { .\SingleLinkedList
Single Linked List Circular (SLC)
=====
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. Keluar
=====
Pilihan : 2
Masukkan Data : 3
Data "3" telah berhasil dimasukkan di bagian belakang.

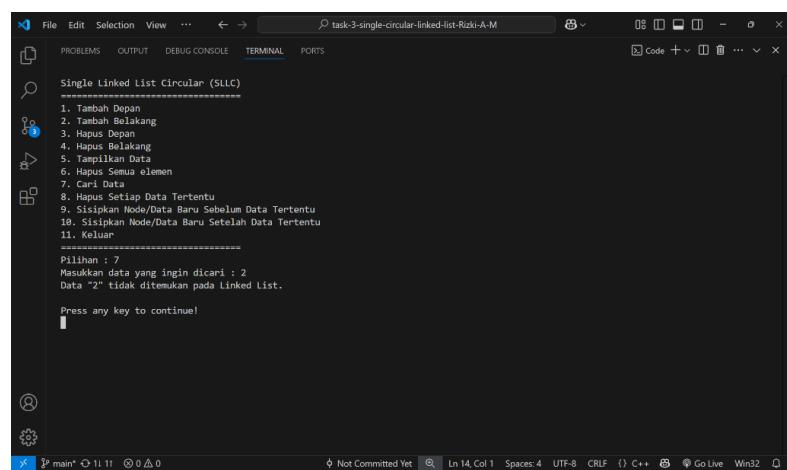
Press any key to continue!
```

Gambar 29. Tampilan Tambah Data dari Belakang



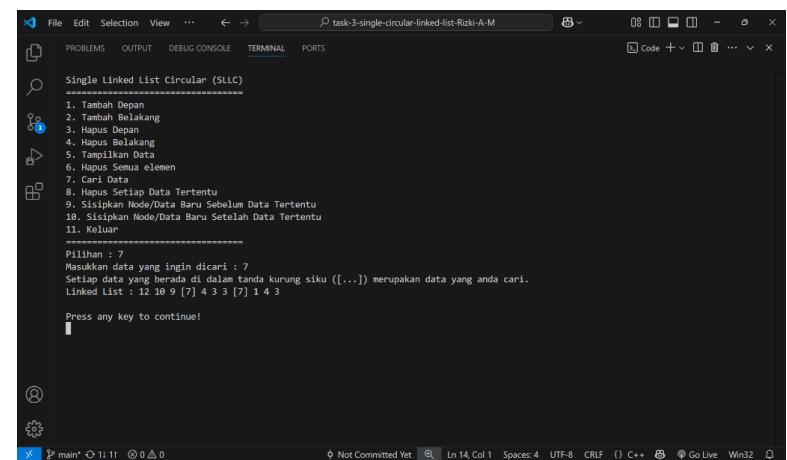
```
task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
12 10 9 7 4 3 3 7 1 4 3
Press any key to continue!
```

Gambar 30. Tampilan Data Setelah Dilakukan Tambah Data Belakang



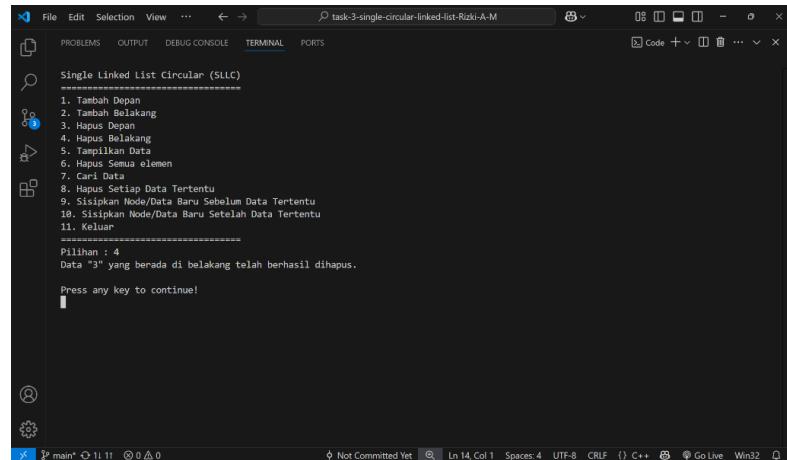
```
task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tampilkan Data
6. Hapus Semua elemen
7. Cari Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 7
Masukkan data yang ingin dicari : 2
Data "2" tidak ditemukan pada Linked List.
Press any key to continue!
```

Gambar 31. Tampilan Cari Data Yang Tidak Ada



```
task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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. Keluar
=====
Pilihan : 7
Masukkan data yang ingin dicari : 7
Setiap data yang anda masukkan dalam tanda kurung siku [...] merupakan data yang anda cari.
Linked List : 12 10 9 [7] 4 3 3 [7] 1 4 3
Press any key to continue!
```

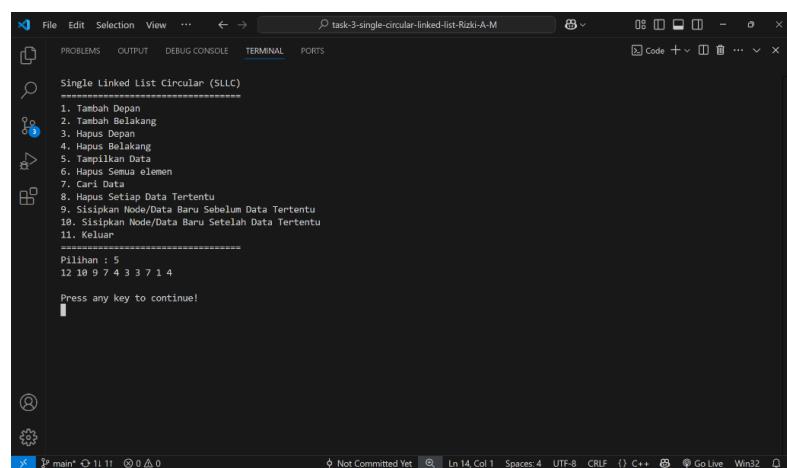
Gambar 32. Tampilan Cari Data Yang Ada



```
task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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. Keluar
=====
Pilihan : 4
Data "3" yang berada di belakang telah berhasil dihapus.

Press any key to continue!
```

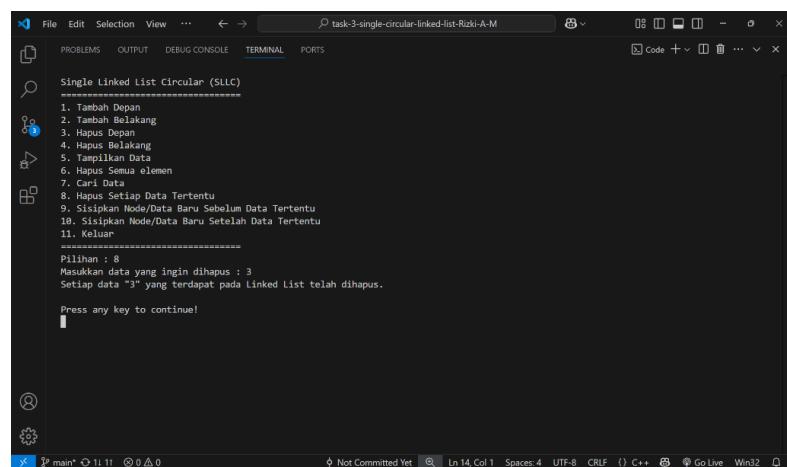
Gambar 33. Tampilan Hapus Data Belakang



```
task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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. Keluar
=====
Pilihan : 5
12 10 9 7 4 3 3 7 1 4

Press any key to continue!
```

Gambar 34. Tampilan Data Setelah Dilakukan Hapus Data Belakang



```
task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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. Keluar
=====
Pilihan : 8
Masukkan data yang ingin dihapus : 3
Setiap data "3" yang terdapat pada Linked List telah dihapus.

Press any key to continue!
```

Gambar 35. Tampilan Hapus Setiap Data Tertentu

```
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. Keluar
=====
Pilihan : 8
12 10 9 7 4 7 1 4

Press any key to continue!
```

Gambar 36. Tampilan Data Setelah Dilakukan Hapus Setiap Data Tertentu

```
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. Keluar
=====
Pilihan : 3
Data "12" yang berada di depan telah berhasil dihapus.

Press any key to continue!
```

Gambar 37. Tampilan Hapus Data Depan

```
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. Keluar
=====
Pilihan : 3
10 9 7 4 7 1 4

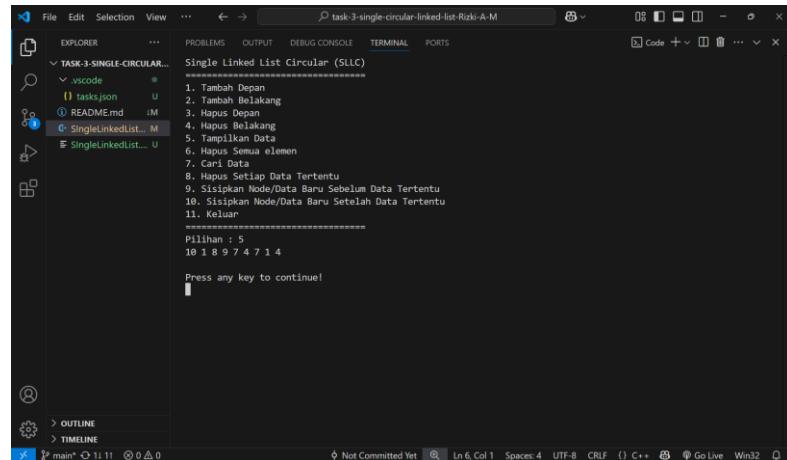
Press any key to continue!
```

Gambar 38. Tampilan Data Setelah Dilakukan Hapus Data Depan

Gambar 39. Tampilan Sisipkan Node Sebelum Data Tertentu

Gambar 40. Tampilan Data Setelah Dilakukan Sisipkan Node Sebelum

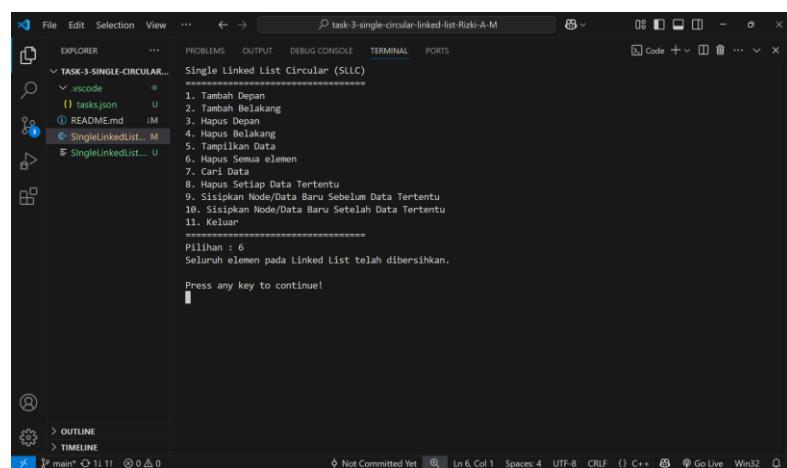
Gambar 41. Tampilan Sisipkan Node Setelah Data Tertentu



```
task-3-single-circular-linked-list Rizki-A-M
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. Keluar
=====
Pilihan : 5
10 1 8 9 7 4 7 1 4

Press any key to continue!
```

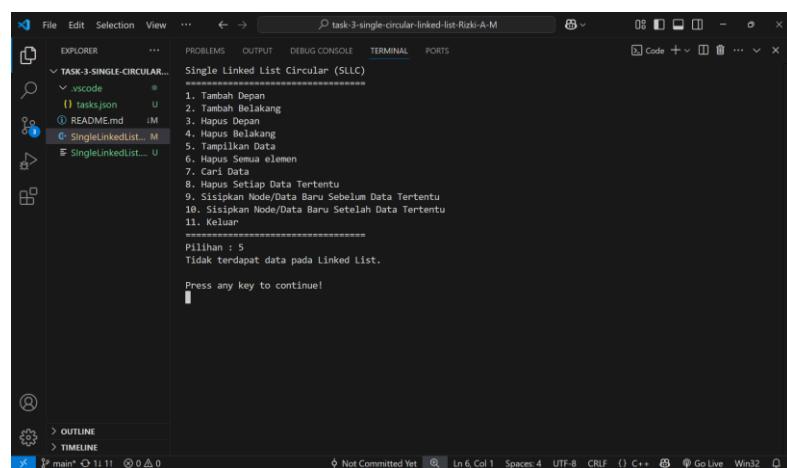
Gambar 42. Tampilan Data Setelah Dilakukan Sisipkan Node Setelah



```
task-3-single-circular-linked-list Rizki-A-M
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. Keluar
=====
Pilihan : 6
Seluruh elemen pada Linked List telah dibersihkan.

Press any key to continue!
```

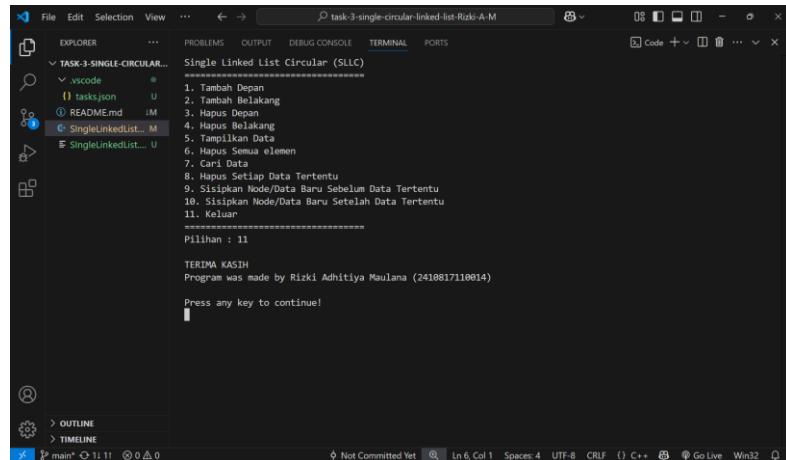
Gambar 43. Tampilan Hapus Semua Elemen



```
task-3-single-circular-linked-list Rizki-A-M
=====
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. Keluar
=====
Pilihan : 5
Tidak terdapat data pada Linked List.

Press any key to continue!
```

Gambar 44. Tampilan Data Setelah Dilakukan Hapus Semua Elemen



Gambar 45. Tampilan Setelah Selesai Menyelesaikan Program

C. Pembahasan

Pada baris [1] sampai [3] terdapat `#include` yang mana digunakan untuk mengakses sebuah file yang diinginkan. `<iostream>` yang ada digunakan untuk input dan output. Kemudian `<conio.h>` digunakan agar menyediakan fungsi fungsi yang berguna ketika ada interaksi langsung dengan keyboard, tanpa perlu menekan Enter. Terus `<stdlib.h>` digunakan untuk fungsi fungsi manajemen memori, konversi angka, kontrol proses, dan lingkungan program.

Pada baris [5] terdapat *using namespace std;* yang mana digunakan untuk menghindari penulisan *std*.

Pada baris [7] sampai [11] terdapat `struct TNode` yang mana digunakan untuk menyimpan elemen-elemen dari linked list, dimana variabel `string data` digunakan untuk menyimpan isi atau informasi dari node tersebut seperti angka yang di input ke dalam list. Kemudian, `TNode *next` digunakan untuk menunjuk ke node berikutnya dalam urutan linked list, node terakhir yang ada dalam urutan linked list akan menunjuk kembali ke node pertama.

Pada baris [13] terdapat `TNode *head, *tail;` yang mana `TNode *head` menunjuk pointer node pertama dan `TNode *tail` menunjuk pointer node terakhir.

Pada baris [15] sampai [17] terdapat *int pil* yang mana digunakan untuk menyimpan variabel Integer atau bilangan bulat. Terus *char pilihan [2]* yang mana digunakan untuk menyimpan variabel character, ditambah array sebagai

batasan input dari user, `string dataBaru` dan `dataDelete` yang digunakan untuk menyimpan variabel string atau katakter.

Pada baris [19] sampai [31] terdapat penamaan fungsi yang akan dimasukkan ke dalam program Linked list.

Pada baris [33] sampai [99] terdapat `int main()` yang mana digunakan untuk menjalankan dan menampilkan menu SLLC. Menu yang akan ditampilkan di dalam sistem ada sebanyak 11 buah, terdiri dari *tambah depan* dan *tambah belakang*, *hapus depan* dan *hapus belakang*, *tampilkan data*, *sapus semua elemen*, *cari data*, *hapus data tertentu*, *sisipkan data sebelum data tertentu* dan *sisipkan data setelah data tertentu*. Setiap pilihan yang ada akan menampilkan tampilan berbeda sesuai dengan fungsi yang ada di dalam `switch case` yang dimasukkan pada program yang akan dijalankan. Terdapat `getch()` untuk menunggu tombol yang ditekan oleh pengguna dan membersihkan layar menggunakan `system("cls")`. Terus program akan terus berjalan selama user tidak memilih pilihan sebelas (11) untuk keluar atau menghentikan program yang ada.

Pada baris [101] sampai [105] terdapat `void Init()` yang mana digunakan untuk menginisialisasikan kondisi awal dari linked list circular. Pointer atau variabel head akan disetting atau diatur dalam keadaan `NULL`, begitu juga dengan pointer atau variabel tail akan disetting atau diatur dalam keadaan `NULL`.

Pada baris [107] sampai [113] terdapat `int Kosong()` yang mana digunakan untuk melakukan pengecekan pada linked list, apakah dalam keadaan kosong atau tidak. Fungsi ini akan mengembalikan nilai 1 apabila linked list dalam keadaan kosong dan 0 apabila tidak dalam keadaan kosong. Fungsi ini akan dipanggil pada saat melakukan operasi penambahan, penghapusan dan menampilkan.

Pada baris [115] sampai [136] terdapat `void TambahDepan()` yang mana digunakan untuk menambahkan node baru ke bagian depan dari linked list. Apabila linked list dalam keadaan kosong, node yang baru ditambahkan akan menjadi head dan tail. Kemudian, apabila linked list dalam keadaan tidak kosong, node yang baru ditambahkan akan menjadi head baru dan node sebelumnya yang jadi head dan tail dalam satu waktu akan menjadi tail.

Pada baris [138] sampai [159] terdapat `void TambahBelakang()` yang mana digunakan untuk menambahkan node baru ke bagian belakang dari linked list. Apabila linked list dalam keadaan kosong, node yang baru ditambahkan akan menjadi tail dan head. Kemudian, apabila linked list dalam keadaan tidak kosong, node yang baru ditambahkan akan menjadi tail baru dan node sebelumnya yang jadi tail dan head dalam satu waktu akan menjadi head.

Pada baris [161] sampai [185] terdapat `void HapusDepan()` yang mana digunakan untuk menghapus node pertama yang terdapat pada linked list. Apabila terdapat satu node saja pada linked list, maka setelah dilakukan hapus depan linked list akan diatur menjadi kosong atau `NULL` menggunakan `fungsi Init()`. Kemudian, apabila terdapat lebih dari satu node yang terdapat pada linked list, node pertama yang dihapus akan digantikan dengan elemen yang ada di setelah sebagai head terbaru dan node pertama baru.

Pada baris [187] sampai [215] terdapat `void HapusBelakang()` yang mana digunakan untuk menghapus node terakhir yang terdapat pada linked list. Apabila terdapat satu node saja pada linked list, maka setelah dilakukan hapus belakang linked list akan diatur menjadi kosong atau `NULL` menggunakan `fungsi Init()`. Kemudian, apabila terdapat lebih dari satu node yang terdapat pada linked list, node terakhir yang dihapus akan digantikan dengan elemen yang ada di sebelumnya sebagai tail terbaru dan node terakhir baru.

Pada baris [217] sampai [236] terdapat `void Tampilkan()` yang mana digunakan untuk menampilkan seluruh isi linked list yang ada, dimulai dari node pertama hingga ke node terakhir. Walapun hanya terdapat satu node di dalam linked list, program akan tetap mencetaknya dan akan berhenti ketika semua elemen yang ada di dalam linked list ditampilkan semua.

Pada baris [238] sampai [259] terdapat `void Reset()` yang mana digunakan untuk menghapus semua node yang ada pada linked list, baik dari node pertama hingga node terakhir. Setelah dilakukan penghapusan untuk semua node yang ada di dalam linked list, kemudian akan dipanggil fungsi `Init()` untuk mengatur linked list ke dalam kondisi kosong.

Pada baris [261] sampai [314] terdapat `void CariData()` yang mana digunakan untuk mencari data tertentu yang ada di dalam linked list. Apabila

ditemukan kesamaan data yang ada di dalam tanda kurung siku dengan yang ada di dalam linked list, maka data yang ada di dalam linked list akan ikut di cetak dalam tanda kurung siku. Namun, apabila tidak ditemukan kesamaan antara data yang ada di dalam tanda kurung siku dengan yang ada di dalam linked list, maka akan muncul tampilan pesan kepada pengguna kalo data A tidak ditemukan.

Pada baris [316] sampai [376] terdapat `void HapusData()` yang mana digunakan untuk menghapus semua node yang ada di dalam linked list, berdasarkan data yang di inputkan oleh pengguna. Fungsi ini akan menghapus node tertentu di dalam linked list ketika pengguna memasukkan sebuah data atau nilai yang ingin di hapus. Ketika selesai memasukkan data yang diinginkan, sistem akan menghapus setiap node yang memiliki kesamaan dengan nilai yang dimasukkan hingga tidak terdapatnya data tersebut di dalam linked list.

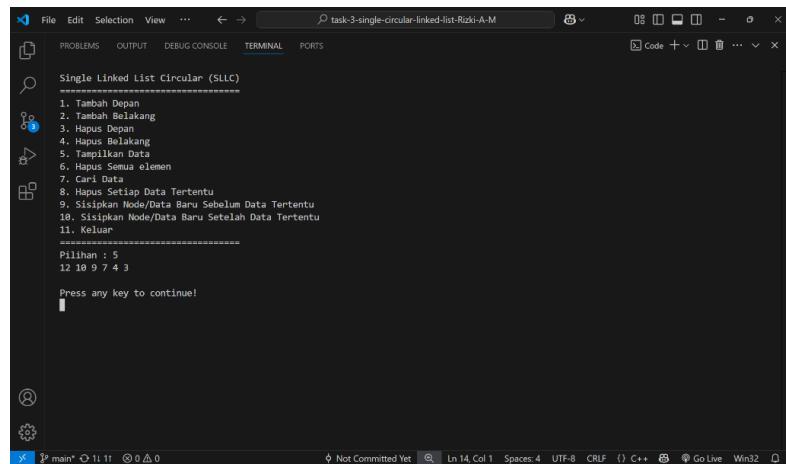
Pada baris [378] sampai [435] terdapat `void SisipkanSebelum()` yang mana digunakan untuk menyisipkan atau memasukkan node baru sebelum node tertentu, sesuai dengan data yang dimasukkan oleh pengguna sebagai data yang akan disisipkan dan data target untuk penyisipan. Sebelum dilakukan penyisipan, terlebih dahulu dilakukan penelusuran pada linked list dari head menuju tail untuk menemukan sebuah node yang menjadi target untuk disisipkan. Setelah target untuk penyisipan ditemukan, maka akan ditambahkan atau disisipkan sebuah node di depannya atau sebelum node tersebut. Apabila node yang menjadi target sisipkan sebelum adalah head dalam linked list maka node yang baru di tambahkan akan menjadi head yang baru.

Pada baris [437] sampai [491] terdapat `void SisipkanSesudah()` yang mana digunakan untuk menyisipkan atau memasukkan node baru setelah node tertentu sesuai dengan data yang dimasukkan oleh pengguna sebagai data yang akan disisipkan dan data target untuk penyisipan. Sebelum dilakukan penyisipan, terlebih dahulu dilakukan penelusuran pada linked list dari head menuju tail untuk menemukan sebuah node yang menjadi target untuk disisipkan. Setelah target untuk penyisipan ditemukan, maka akan ditambahkan atau disisipkan sebuah node di belakang atau setelah node tersebut. Apabila node yang menjadi target sisipkan setelah adalah tail dalam linked list maka node yang baru di tambahkan akan menjadi tail yang baru.

SOAL 2

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

A. Output Program



```
task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tambah 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. Keluar
=====
Pilihan : 5
12 10 9 7 4 3

Press any key to continue!
```

Gambar 46. Tampilan Nilai yang Diinput dari Depan

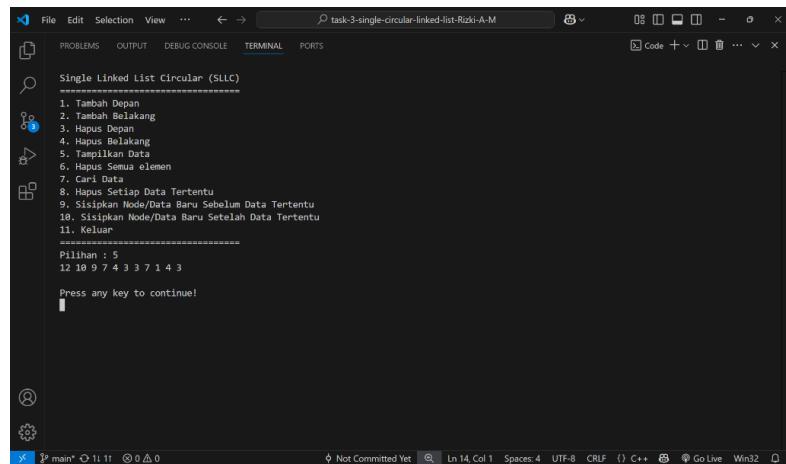
B. Pembahasan

Ketika data dimasukkan ke dalam Linked List melalui bagian depan, dengan urutan masuk mulai dari 3, 4, 7, 9, 10, dan 12. Angka 3 yang pertama kali dimasukkan akan muncul di sebelah kiri ketika di tampilkan. Data-data yang muncul setelah data pertama ditambahkan akan menjadi head dari Linked List dan data sebelumnya akan digeser ke samping kiri atau ke belakang.

SOAL 3

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

A. Output Program



```
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Tambah 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. Keluar
=====
Pilihan : 5
12 10 9 7 4 3 3 7 1 4 3

Press any key to continue!
```

Gambar 47. Tampilan Nilai yang Diinput dari Belakang

B. Pembahasan

Ketika data dimasukkan ke dalam Linked List melalui bagian belakang, dengan urutan masuk mulai dari 3, 7, 1, 4, dan 3. Angka 3 yang pertama kali dimasukkan akan muncul di sebelah kiri ketika di tampilkan. Data-data yang muncul setelah data pertama ditambahkan akan menjadi tail dari Linked List dan tampilkan dari Linked List akan sama seperti saat kita memasukkan data ke dalam Linked List.

SOAL 4

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

A. Output Program

Gambar 48. Pencarian Nilai 2 pada Linked List

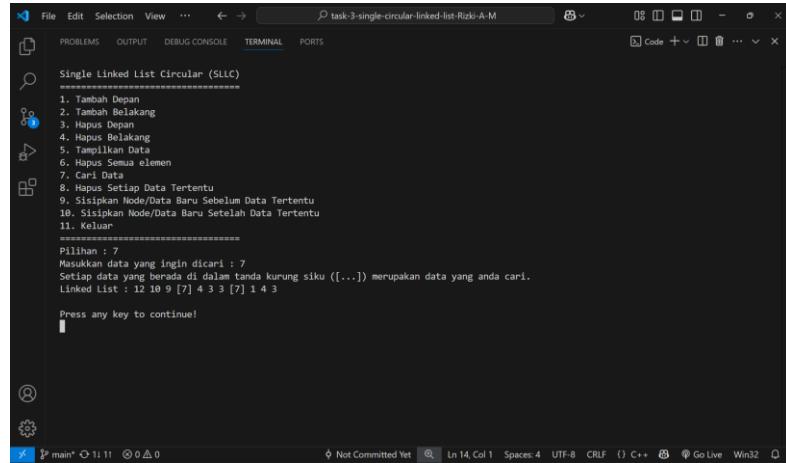
B. Pembahasan

Berdasarkan data yang telah dimasukkan pada soal 2 dan soal 3, ketika dilakukan sebuah pencarian untuk nilai 2. Sistem akan memberikan informasi bahwa nilai tersebut tidak terdapat atau tidak ditemukan pada Linked List.

SOAL 5

Coba cari angka 7 dan screenshoot hasilnya !

A. Output Program



The screenshot shows a terminal window with the following text:

```
task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Empat 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. Keluar
=====

Pilihan : 7
Masukkan data yang ingin dicari : 7
Setiap data yang berada di dalam tanda kurung siku [...] merupakan data yang anda cari.
Linked List : 12 10 9 [7] 4 5 3 [7] 1 4 3

Press any key to continue!
```

Gambar 49. Pencarian Nilai 7 pada Linked List

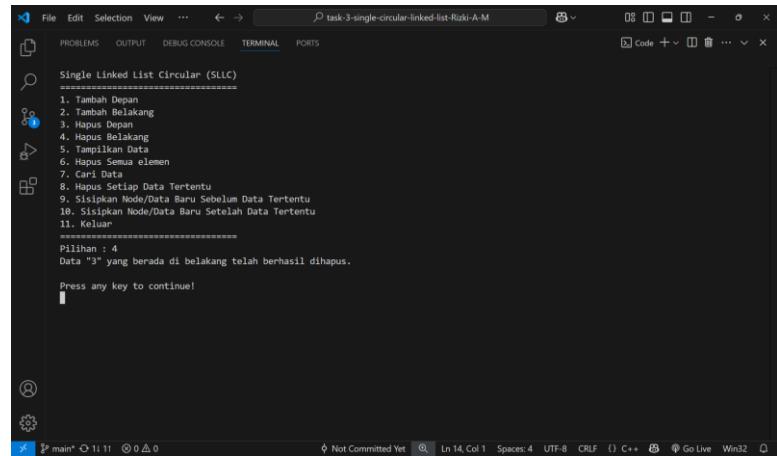
B. Pembahasan

Berdasarkan data yang telah dimasukkan pada soal 2 dan soal 3, ketika dilakukan sebuah pencarian untuk nilai 7. Sistem akan memberikan informasi bahwa nilai yang dicari akan ditandai dengan kurung siku [...] pada Linked List.

SOAL 6

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

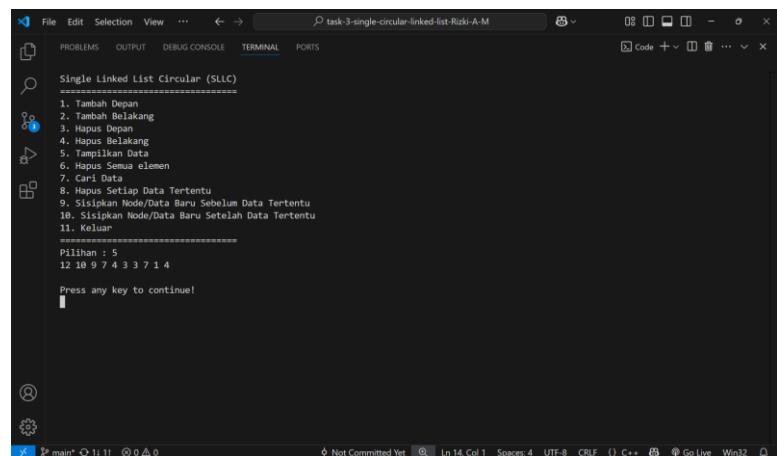
A. Output Program



```
task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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. Carl Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 4
Data "3" yang berada di belakang telah berhasil dihapus.

Press any key to continue!
```

Gambar 50. Menghapus Data dari Belakang Linked List



```
task-3-single-circular-linked-list-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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. Carl Data
8. Hapus Setiap Data Tertentu
9. Sisipkan Node/Data Baru Sebelum Data Tertentu
10. Sisipkan Node/Data Baru Setelah Data Tertentu
11. Keluar
=====
Pilihan : 5
12 10 9 7 4 3 3 7 1 4

Press any key to continue!
```

Gambar 51. Tampilan Linked List Setelah Dihapus

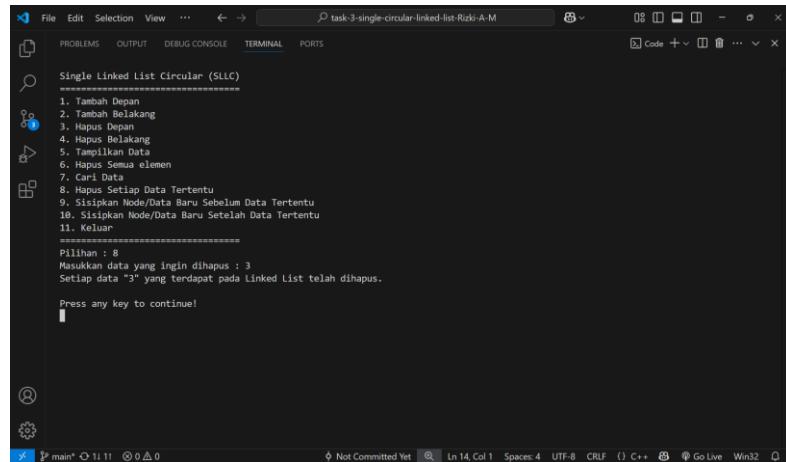
B. Pembahasan

Berdasarkan data yang telah dimasukkan pada soal 2 dan soal 3, ketika dilakukan sebuah penghapusan data untuk bagian belakang atau tail. Nilai 3 yang terletak di bagian paling belakang pada Linked List (tail) akan dihapuskan dan ketika Linked List di tampilkan nilai yang ada disebelah kiri dari nilai yang dihapus akan menjadi tail yang baru.

SOAL 7

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

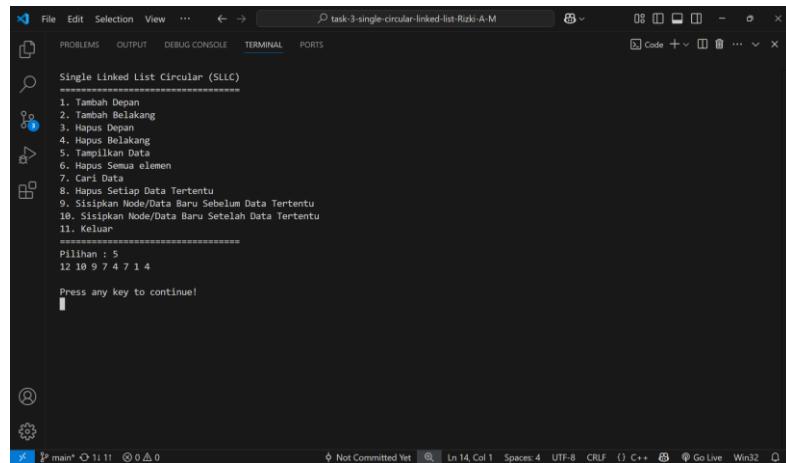
A. Output Program



```
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Sisipkan 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. Keluar
=====
Pilihan : 8
Masukkan data yang ingin dihapus : 3
Setiap data '3' yang terdapat pada Linked List telah dihapus.

Press any key to continue!
```

Gambar 52. Menghapus Setiap Nilai 3 yang Ada di Linked List



```
Single Linked List Circular (SLLC)
=====
1. Tambah Depan
2. Tambah Belakang
3. Hapus Depan
4. Hapus Belakang
5. Sisipkan 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. Keluar
=====
Pilihan : 8
12 10 9 7 4 7 1 4

Press any key to continue!
```

Gambar 53. Tampilan Linked List Setelah Setiap Nilai 3 Dihapus

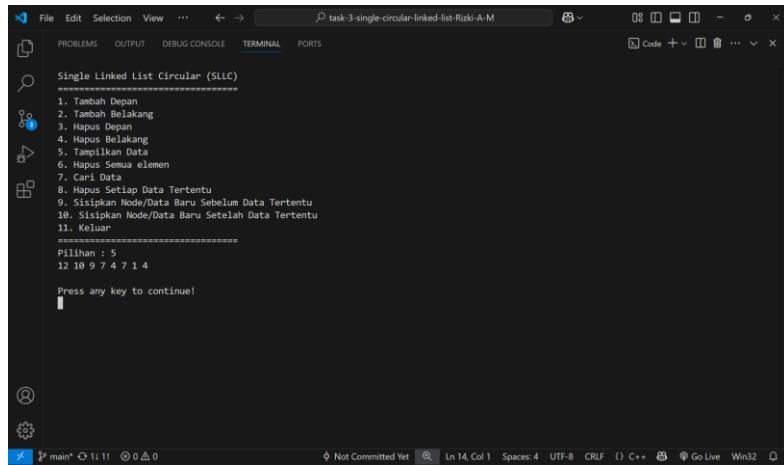
B. Pembahasan

Berdasarkan data yang telah dimasukkan pada soal 2 dan soal 3, ketika dilakukan sebuah penghapusan untuk setiap Nilai 3 yang ada pada Linked List. Nilai 3 yang terdapat pada Linked List akan dihapuskan semua dan ketika Linked List di tampilkan tidak akan terdapat nilai 3 di dalamnya.

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. Tambah Semua 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. Keluar
=====
Pilihan : 5
12 10 9 7 4 7 1 4

Press any key to continue!
```

Gambar 54. Tampilan Isi dari Linked List

B. Pembahasan

Berdasarkan data yang telah dimasukkan pada soal 2 dan soal 3, kemudian dilakukannya penghapusan pada soal 6 dan soal 7. Data yang berada di paling depan atau di samping kiri akan menjadi head yaitu 12, dapat juga dibuktikan dengan melakukan sebuah operasi penambahan atau penghapusan data dari depan yang mana akan mempengaruhi data atau nilai dari 12 itu sendiri. Apabila ditambah data 1 pada Linked List dari depan data yang baru ditambahkan tersebut yang akan menjadi head. Begitu juga dengan penghapusan data atau nilai dari 12, maka data atau nilai yang ada di belakangnya yang akan menjadi head baru. Sedangkan data yang berada di paling belakang atau di samping kanan akan menjadi tail yaitu 4, dapat juga dibuktikan dengan melakukan sebuah operasi penambahan atau penghapusan data dari belakang yang mana akan mempengaruhi data atau nilai dari 4 itu sendiri. Apabila ditambah data 1 pada Linked List dari belakang data yang baru ditambahkan tersebut yang akan menjadi tail. Begitu juga dengan penghapusan data atau nilai dari 4, maka data atau nilai yang ada di depannya yang akan menjadi tail baru.

SOAL 9

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

A. Pembahasan

Apabila baris ke-103 dan 104 dihapuskan dari program yang mana pada baris itu terdapat `if(head == NULL) return 1; dan else return 0;` di dalamnya. Maka akan fungsi pengecekan untuk mengetahui apakah Linked List kosong atau berisi tidak bisa dilakukan di dalam program. Tanpa kedua baris kode tersebut, program masih bisa berjalan, namun berisiko mengalami kesalahan atau perilaku yang tidak terduga, seperti penggunaan garbage value atau pengaksesan memori yang tidak valid, yang dapat menyebabkan crash atau error lainnya.

SOAL 10

Jelaskan apa itu variabel head dan tail pada SLLC!

A. Pembahasan

Pada Single Linked List Circular (SLLC), variabel head merupakan pointer yang menunjuk ke node pertama dalam Linked List, atau dengan kata lain, variabel head berfungsi sebagai tempat pertama data akan ditelusuri. Sedangkan variabel tail merupakan pointer yang menunjuk ke node terakhir dalam Linked List, atau tempat terakhir data akan ditelusuri.

Karena SLLC bersifat melingkar (*circular*), maka node terakhir (*tail*) akan menunjuk kembali ke node pertama (*head*), yang mana akan menjadikannya sebuah struktur yang terus berputar. Singkatnya, tail sebagai node terakhir akan menunjuk kembali ke head, sehingga memungkinkan traversal list secara terus-menerus tanpa ada akhir.

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 pilhan[1];
17 string databaru;
18
19 void init();
20 void initH();
21 int isEmptyH();
22 int isEmpty();
23
24 void tambahDepan();
25 void tambahDepanH();
26 void tambahBelakang();
27 void tambahBelakangH();
28 void hapusDepan();
29 void hapusDepanH();
30 void hapusBelakang();
31 void hapusBelakangH();
32 void tampilan();
33 void tampilanH();
34 void clearH();
35 void clearH();
36
37 int main()
38 {
39     menu:
40     cout<<"-----Double Linked List Non Circular (DLINC)"<<endl;
41     cout<<"-----"<<endl;
42     cout<<"Silahkan pilih program DLINC yang ingin dijalankan!"<<endl;
43     cout<<"1. DLINC dengan Head"<<endl;
44     cout<<"2. DLINC dengan Head dan Tail"<<endl;
45     cout<<"3. Pilhan : ";
46     cout<<"Pilhan : ";
47     cin>>menu;
48     system("cls");
49
50     if(menu==1){
51         do {
52             cout<<"Double Linked List Non Circular (DLINC) (Head)"<<endl;
53             cout<<"-----"<<endl;
54             cout<<"1. Tambah Depan"<<endl;
55             cout<<"2. Tambah Belakang"<<endl;
56             cout<<"3. Tampilkan Data"<<endl;
57             cout<<"4. Hapus Depan"<<endl;
58             cout<<"5. Hapus Belakang"<<endl;
59             cout<<"6. Reset"<<endl;
60             cout<<"7. Kembali ke Menu"<<endl;
61             cout<<"Pilihan : ";
62             cin>>pilhan;
63             pil=atoi(pilhan);
64
65             switch(pil) {
66                 case 1:
67                     tambahDepan();
68                     break;
69                 case 2:
70                     tambahBelakang();
71                     break;
72                 case 3:
73                     tampilan();
74                     break;
75                 case 4:
76                     hapusDepan();
77                     break;
78                 case 5:
79                     hapusBelakang();
80                     break;
81                 case 6:
82                     clearH();
83                     break;
84                 default:
85                     system("cls");
86                     goto menu;
87             }
88             cout<<"\nPress any key to continue"<<endl;
89             getch();
90             system("cls");
91
92         } while (pil<>);
93     } else if(menu==2){
94         do {
```

```

95     cout<<"Double Linked List Non Circular (DLNC) (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. Keluar ke Menu" <<endl;
104    cout<<"Pilihan : ";
105    cin>>pilihan;
106    pil=atoi(pilihan);
107
108    switch(pil) {
109    case 1:
110        tambahDepan();
111        break;
112    case 2:
113        tambahBelakang();
114        break;
115    case 3:
116        tampilkan();
117        break;
118    case 4:
119        hapusDepan();
120        break;
121    case 5:
122        hapusBelakang();
123        break;
124    case 6:
125        clear();
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<<"TERIMA KASIH"<<endl;
139     cout<<"Program was made by Nama (NIM). "<<endl;
140 }
141
142
143 void initH(){
144     head = NULL;
145 }
146
147 void initT(){
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 isEmptyT(){
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 "<<dataBaru<<" berhasil dimasukkan di bagian depan." ;
178 }
179
180 void tambahDepanT() {
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(isEmptyH() == 1) {
189         head = baru;
190         tail = baru;
191     } else {
192         baru->next = head;
193         head->prev = baru;
194         head = baru;
195     }
196     cout << "Data "<<dataBaru<<" 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 "<<dataBaru<<" berhasil dimasukkan di bagian belakang." ;
218 }
219

```

```

220 void tambahBelakangH() {
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(isEmptyH() == 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 tampilanH() {
240     TNode *bantu;
241     bantu = head;
242     if(isEmptyH() == 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 tampilanHT() {
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(isEmptyH() == 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
287         if(head->next != NULL) {
288             head = head->next;
289             head->prev = NULL;
290         } else {
291             inith();
292         }
293         delete hapus;
294         cout<<"Data \"<<data<<\"\" yang berada di depan telah berhasil dihapus.";
295     } else cout<<"Tidak terdapat data pada Linked List";
296 }
297
298 void hapusBelakangH() {
299     TNode *hapus;
300     string data;
301     if(isEmptyH() == 0) {
302         hapus = head;
303         while(hapus->next != NULL){
304             hapus = hapus->next;
305         }
306         data = hapus->data;
307         if(head->next != NULL) {
308             hapus->next->prev = NULL;
309         } else {
310             inith();
311         }
312         delete hapus;
313         cout<<"Data \"<<data<<\"\" yang berada di belakang telah berhasil dihapus.";
314     } else cout<<"Tidak terdapat data pada Linked List";
315 }
316
317 void hapusBelakangHT() {
318     TNode *hapus;
319     string data;
320     if(isEmptyHT() == 0) {
321         hapus = tail;
322         data = hapus->data;
323         if(head->next != NULL) {
324             tail = tail->prev;
325             tail->next = NULL;
326         } else {
327             inith();
328         }
329         delete hapus;
330         cout<<"Data \"<<data<<\"\" yang berada di belakang telah berhasil dihapus.";
331     } else cout<<"Tidak terdapat data pada Linked List";
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 55. Soal 5 Modul 4

A. Source Code

Tabel 7. Source Code Soal 1 Modul 4

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 dataBaru;
18	
19	void initH();
20	void initHT();
21	int isEmptyH();


```

50      do {
51          cout<<"Double      Linked      List      Non
52 Circular (DLLNC) (head)"<<endl;
53
54      cout<<"=====
55      ====="<<endl;
56      cout<<"1. Tambah Depan"<<endl;
57      cout<<"2. Tambah Belakang"<<endl;
58      cout<<"3. Tampilkan Data"<<endl;
59      cout<<"4. Hapus Depan"<<endl;
60      cout<<"5. Hapus Belakang"<<endl;
61      cout<<"6. Reset"<<endl;
62      cout<<"7. Kembali Ke Menu"<<endl;
63      cout<<"Pilihan : "<<endl;
64      cin>>pilihan;
65      pil=atoi(pilihan);
66
67      switch (pil){
68      case 1:
69          tambahDepanH();
70          break;
71      case 2:
72          tambahBelakangH();
73          break;
74      case 3:
75          tampilkanH();
76          break;
77      case 4:
78          hapusDepanH();

```

```

79             break;
80
81             case 6:
82
83             clearH();
84
85             break;
86
87             default:
88
89             system("cls");
90
91             goto menu;
92
93             }
94
95             cout<<"\npress      any      key      to
96             continue"<<endl;
97
98             getch();
99
100            system("cls");
101
102            } while (pil<7);
103
104            } else if(menu==2) {
105
106            do {
107
108                cout<<"Double      Linked      List      Non
109                Circular (DLLNC) (head dan Tail)"<<endl;
110
111                cout<<"=====
112                ====="<<endl;
113
114                cout<<"1. Tambah Depan"<<endl;
115                cout<<"2. Tambah Belakang"<<endl;
116                cout<<"3. Tampilkan Data"<<endl;
117                cout<<"4. Hapus Depan"<<endl;
118                cout<<"5. Hapus Belakang"<<endl;
119                cout<<"6. Reset"<<endl;
120                cout<<"7. Kembali Ke Menu"<<endl;
121
122                cout<<"Pilihan : "<<endl;
123
124                cin>>pilihan;
125
126                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                 tampilanHT();
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
133             continue"<<endl;
134             getch();
135             system("cls");
136             } while (pil<7);
137         } else {

```

```

138         cout<<"\nTERIMA KASIH"<<endl;
139         cout<<"Program was made by Rizki Adhitiya
140             Maulana (2410817110014)"<<endl;
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<<"Masukan Data : ";
164     cin>>dataBaru;
165     Tnode *baru;
166     baru = new Tnode;
167     baru->data = dataBaru;
168     baru->next = NULL;

```

```

169     baru->prev = NULL;
170     if(isEmptyH() == 1) {
171         head = baru;
172     } else {
173         baru->next = head;
174         head->prev = baru;
175         head = baru;
176     }
177     cout<<"Data     \"<<dataBaru<<\" berhasil
178     dimasukan di bagian depan.";
179 }
180 void tambahDepanHT() {
181     cout<<"Masukan 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     \"<<dataBaru<<\" berhasil
197     dimasukan di bagian depan.";
198 }

```

```

199 void tambahBelakangH() {
200     cout<<"Masukan 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 " <<dataBaru<<" " berhasil
218     dimasukan di bagian belakang.";
219 }
220 void tambahBelakangHT() {
221     cout<<"Masukan 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
237     dimasukan di bagian belakang.";
238 }
239 void tampilanH() {
240     Tnode *bantu;
241     bantu = head;
242     if(isEmptyH() == 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
249     List";
250 }
251 void tampilanHT() {
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
261 List";
262
263 void hapusDepanH() {
264     Tnode *hapus;
265     string data;
266     if(isEmptyH() == 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<<"\"     berhasil
277 dihapus dari bagian depan."<<endl;
278     } else cout<<"Tidak ada data pada Linked
279 List"<<endl;
280 }
281 void hapusDepanHT() {
282     Tnode *hapus;
283     string data;
284     if(isEmptyHT() == 0) {
285         hapus = head;
286         data = hapus->data;
287         if(head->next != NULL) {
288             head = head->next;
289             head->prev = NULL;

```

```

289     } else {
290         initHT();
291     }
292     delete hapus;
293     cout<<"Data     \\""<<<data<<"\\"    berhasil
294     dihapus dari bagian depan."<<endl;
295 } else cout<<"Tidak ada data pada Linked
296 List"<<endl;
297 }
298
299 void hapusBelakangH() {
300     Tnode *hapus;
301     string data;
302     if(isEmptyH() == 0) {
303         hapus = head;
304         while(hapus->next != NULL) {
305             hapus = hapus->next;
306         }
307         data = hapus->data;
308         if(head->next != NULL) {
309             hapus->prev->next = NULL;
310         } else {
311             initH();
312         }
313         delete hapus;
314         cout<<"Data     \\""<<<data<<"\\"    berhasil
315         dihapus dari bagian belakang."<<endl;
316     } else cout<<"Tidak ada data pada Linked
317 List"<<endl;
318 }
319
320 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<<"\"     berhasil
dihapus dari bagian belakang."<<endl;
330     } else cout<<"Tidak ada data pada Linked
List"<<endl;
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"<<endl;
343 }
344
345 void clearHT() {

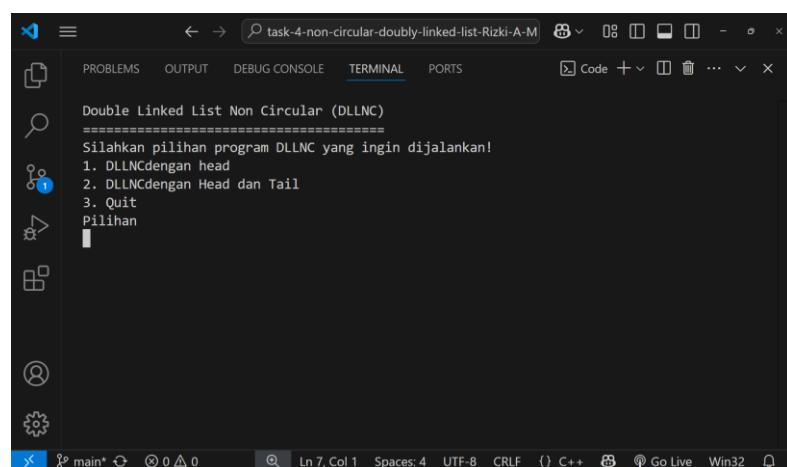
```

```

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
355     dibersihkan"<<endl;
}

```

B. Output Program

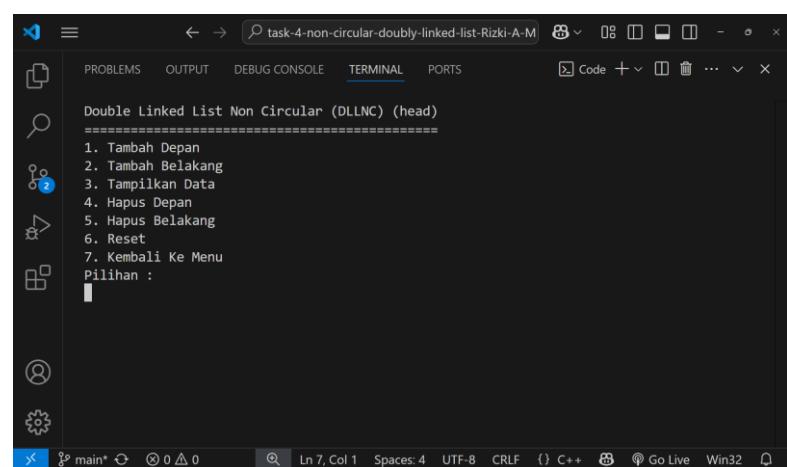


```

Double Linked List Non Circular (DLLNC)
=====
Silahkan pilihkan pilihan program DLLNC yang ingin dijalankan!
1. DLLNCdengan head
2. DLLNCdengan Head dan Tail
3. Quit
Pilihan

```

Gambar 56. Tampilan Menu Double Linked List Non Circular

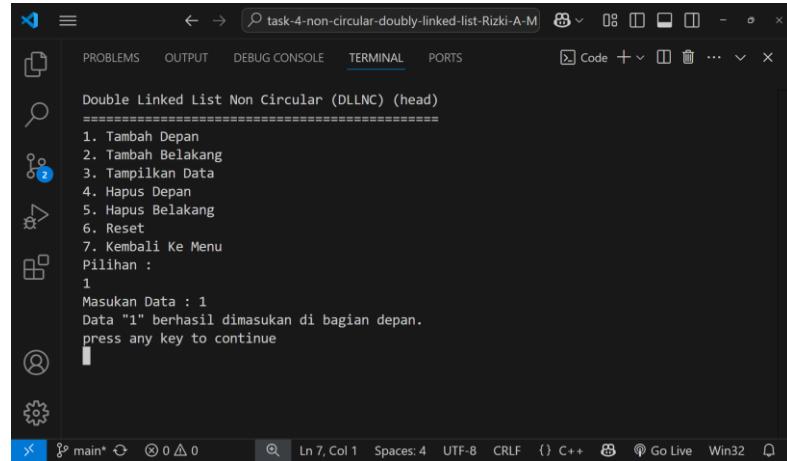


```

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 :

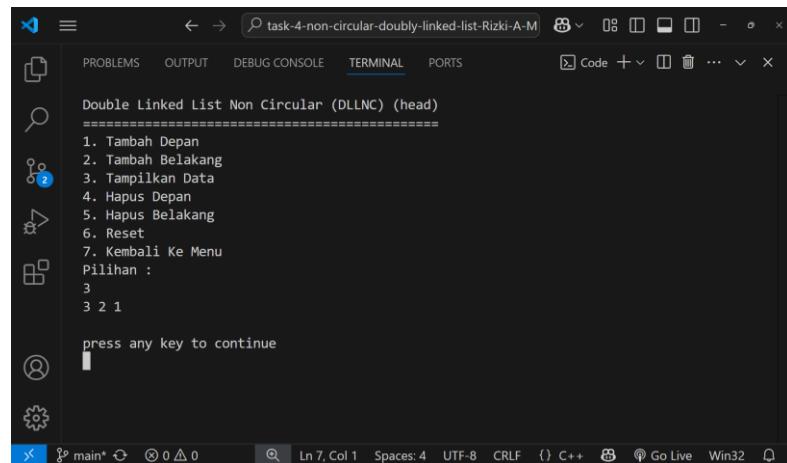
```

Gambar 57. Masuk Ke Tampilan Menu Head DLLNC



```
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 :
1
Masukan Data : 1
Data "1" berhasil dimasukan di bagian depan.
press any key to continue
```

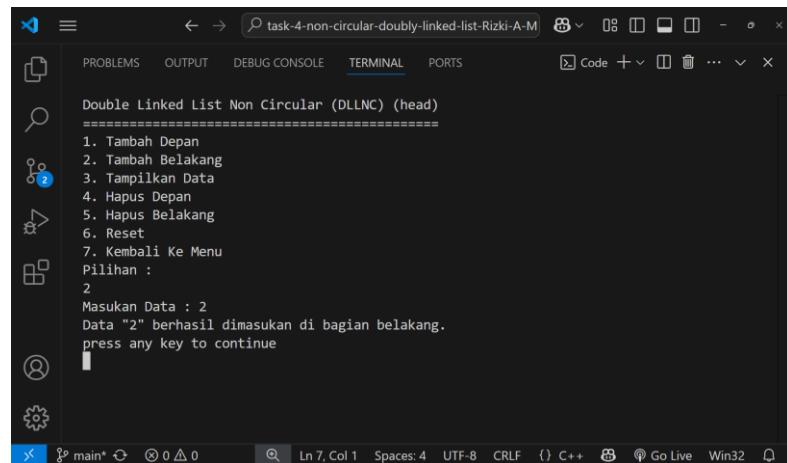
Gambar 58. Tambah Data Dari Depan



```
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
3 2 1

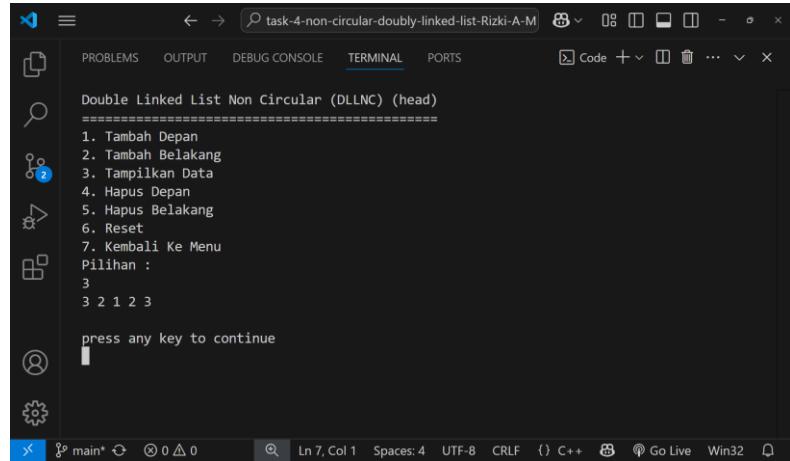
press any key to continue
```

Gambar 59. Tampilan Data Setelah Dilakukan Tambah Depan



```
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 :
2
Masukan Data : 2
Data "2" berhasil dimasukan di bagian belakang.
press any key to continue
```

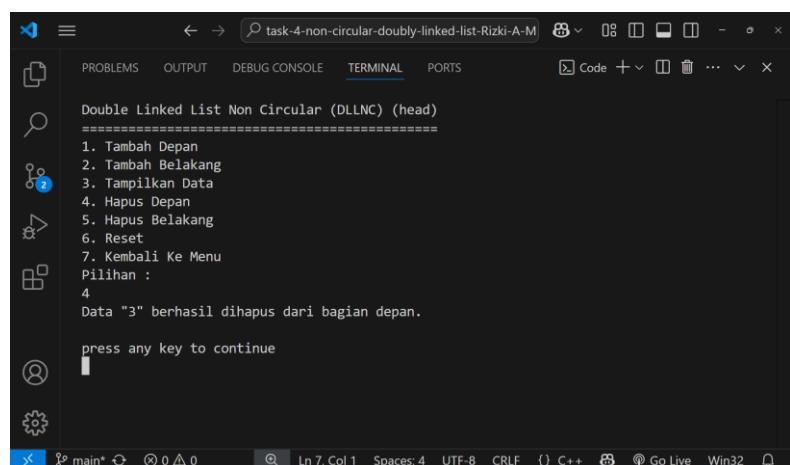
Gambar 60. Tambah Data Dari Belakang



```
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
3 2 1 2 3

press any key to continue
```

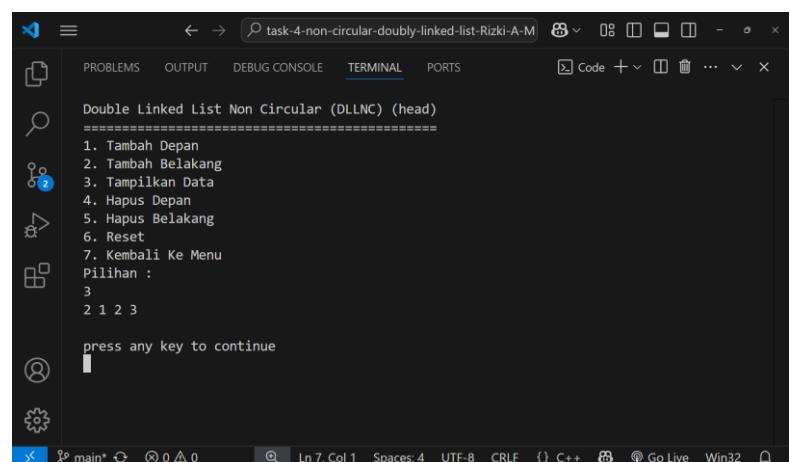
Gambar 61. Tampilan Data Setelah Dilakukan Tambah Belakang



```
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 :
4
Data "3" berhasil dihapus dari bagian depan.

press any key to continue
```

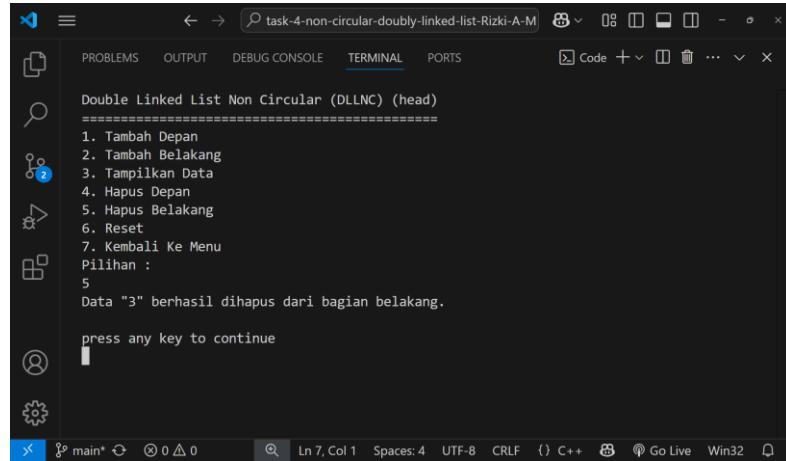
Gambar 62. Hapus Data Dari Depan



```
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
2 1 2 3

press any key to continue
```

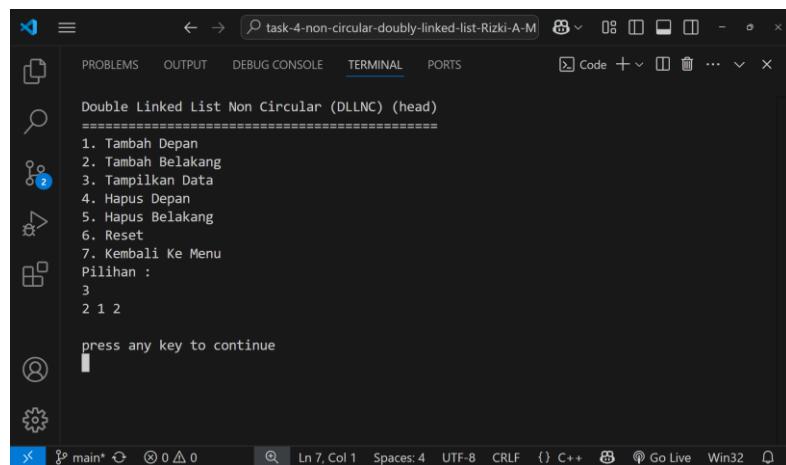
Gambar 63. Tampilan Data Setelah Dilakukan Hapus Depan



```
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 :
5
Data "3" berhasil dihapus dari bagian belakang.

press any key to continue
```

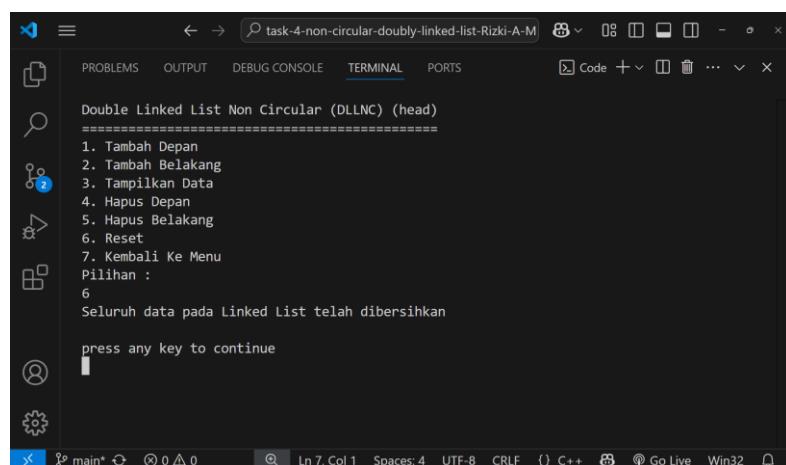
Gambar 64. Hapus Data Dari Belakang



```
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
2 1 2

press any key to continue
```

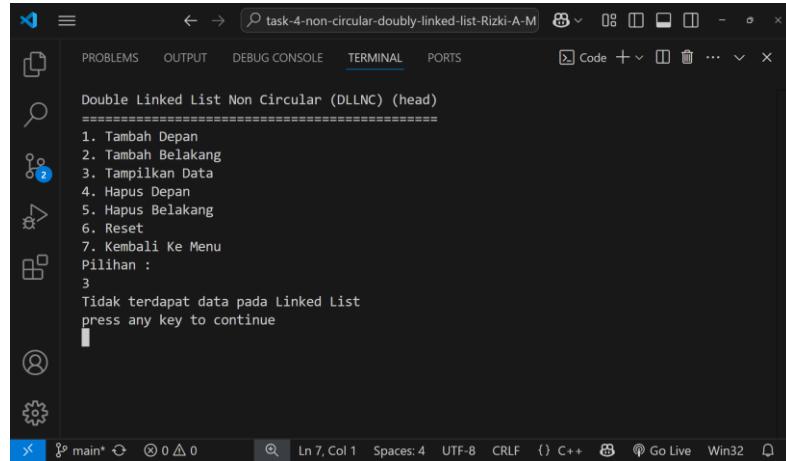
Gambar 65. Tampilan Data Setelah Dilakukan Hapus Belakang



```
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 :
6
Seluruh data pada Linked List telah dibersihkan

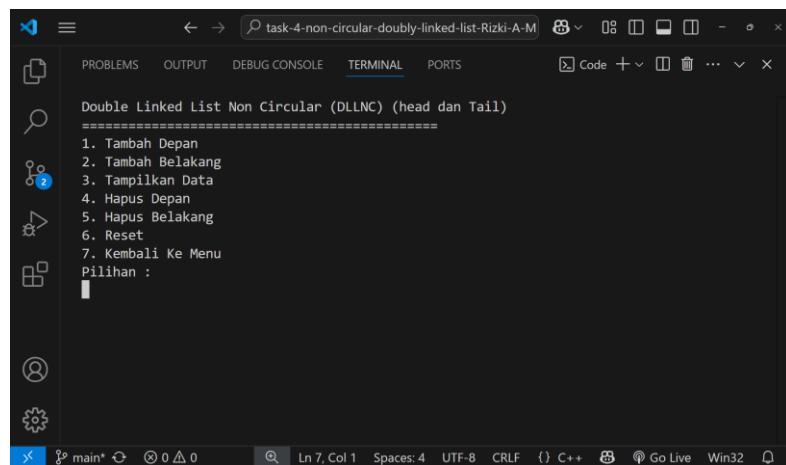
press any key to continue
```

Gambar 66. Reset Data Yang Ada



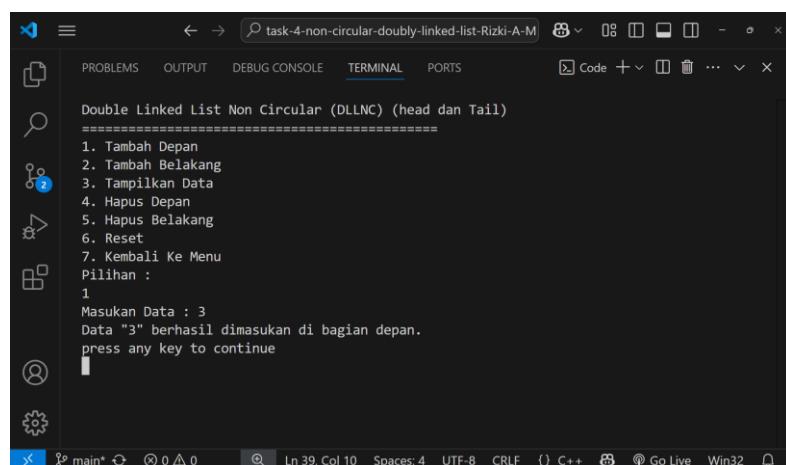
```
Double Linked List Non Circular (DLLNC) (head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
3
Tidak terdapat data pada Linked List
press any key to continue
```

Gambar 67. Tampilan Data Setelah Dilakukan Reset Data



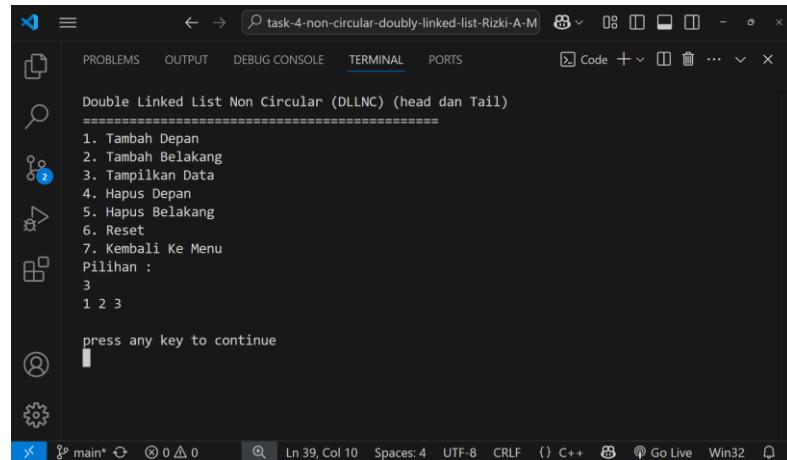
```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
```

Gambar 68. Masuk Ke Tampilan Menu Head Dan Tail DLLNC



```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
1
Masukan Data : 3
Data "3" berhasil dimasukan di bagian depan.
press any key to continue
```

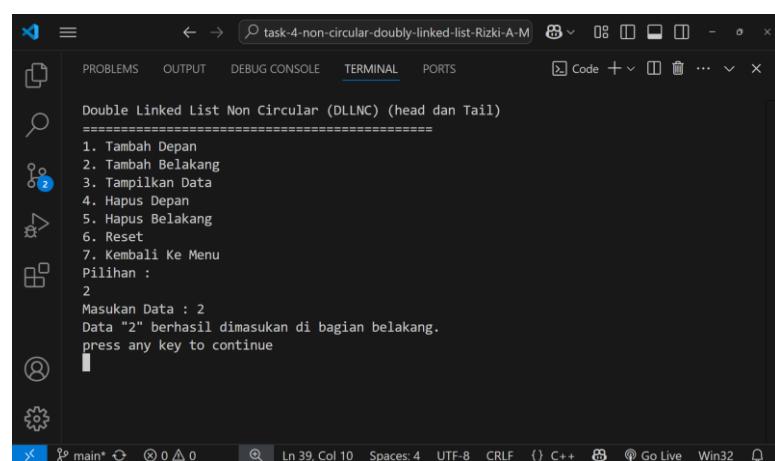
Gambar 69. Tambah Data Dari Depan



```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
3
1 2 3

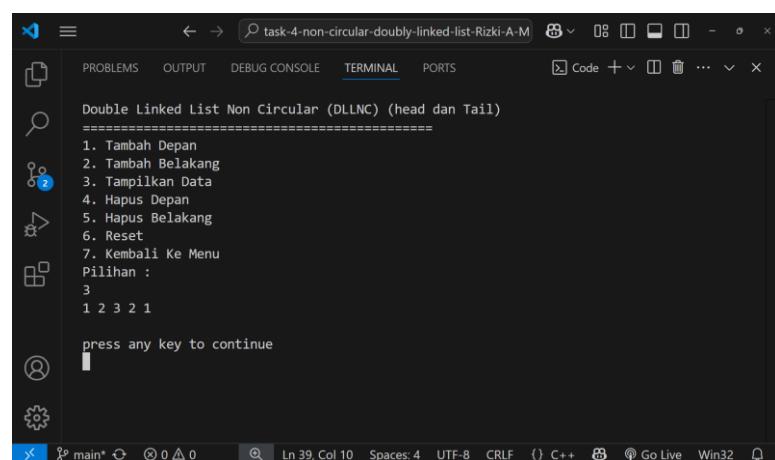
press any key to continue
```

Gambar 70. Tampilan Data Setelah Dilakukan Tambah Depan



```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
2
Masukan Data : 2
Data "2" berhasil dimasukan di bagian belakang.
press any key to continue
```

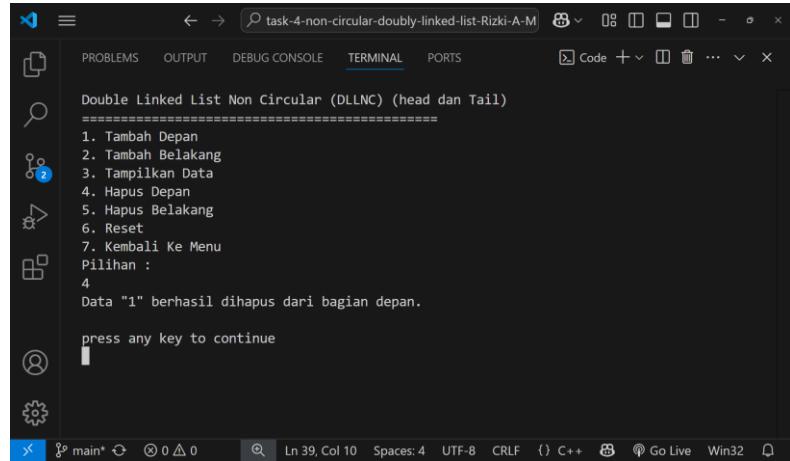
Gambar 71. Tambah Data Dari Belakang



```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
3
1 2 3 2 1

press any key to continue
```

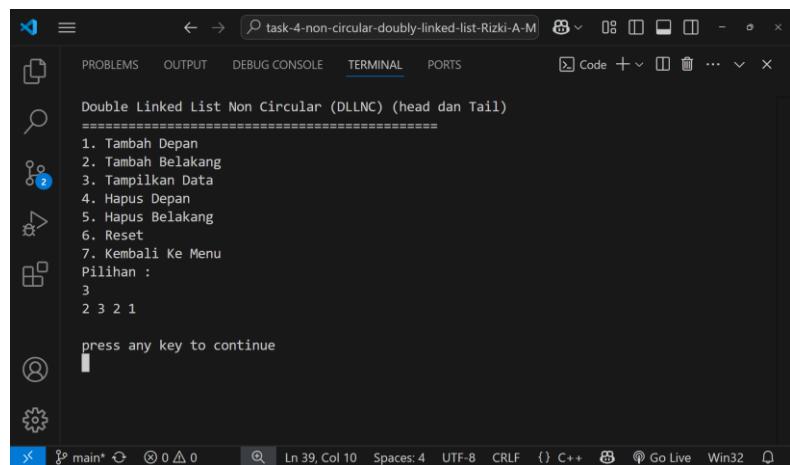
Gambar 72. Tampilan Data Setelah Dilakukan Tambah Belakang



```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
4
Data "1" berhasil dihapus dari bagian depan.

press any key to continue
```

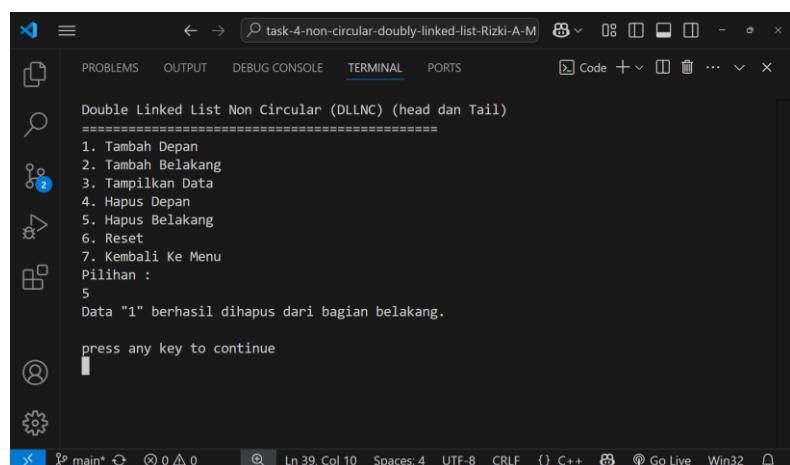
Gambar 73. Hapus Data Dari Depan



```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
3
2 3 2 1

press any key to continue
```

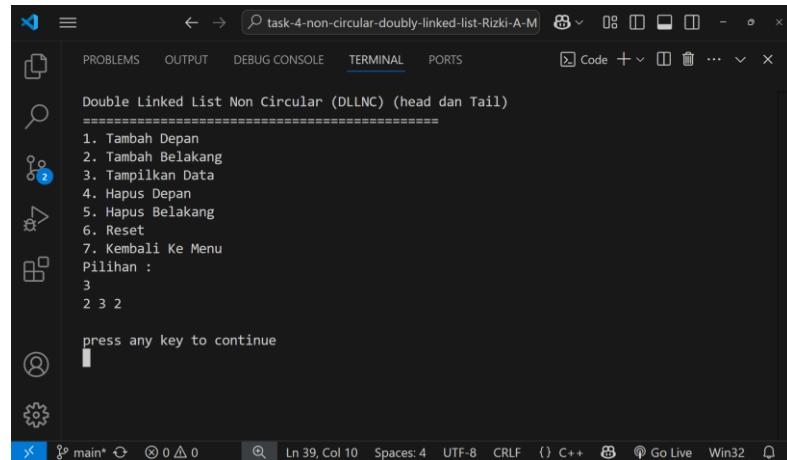
Gambar 74. Tampilan Data Setelah Dilakukan Hapus Depan



```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
5
Data "1" berhasil dihapus dari bagian belakang.

press any key to continue
```

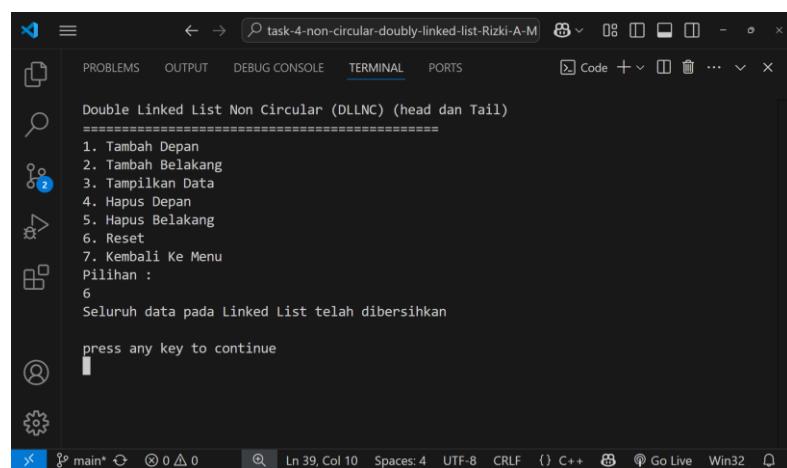
Gambar 75. Hapus Data Dari Belakang



```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
3
2 3 2

press any key to continue
```

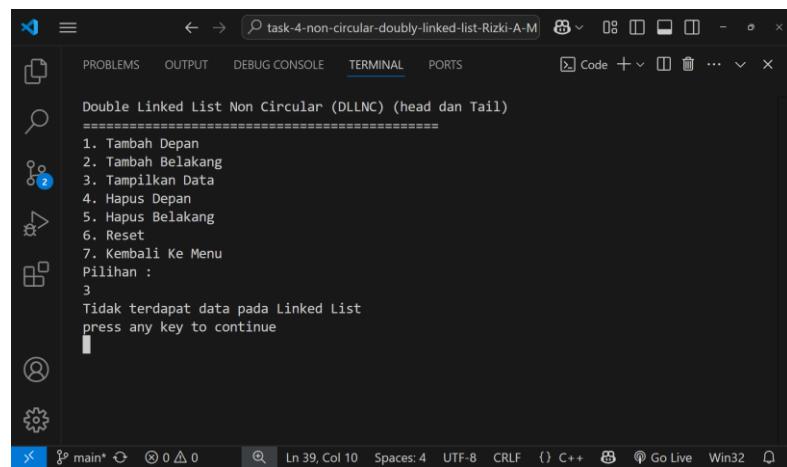
Gambar 76. Tampilan Data Setelah Dilakukan Hapus Belakang



```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
6
Seluruh data pada Linked List telah dibersihkan

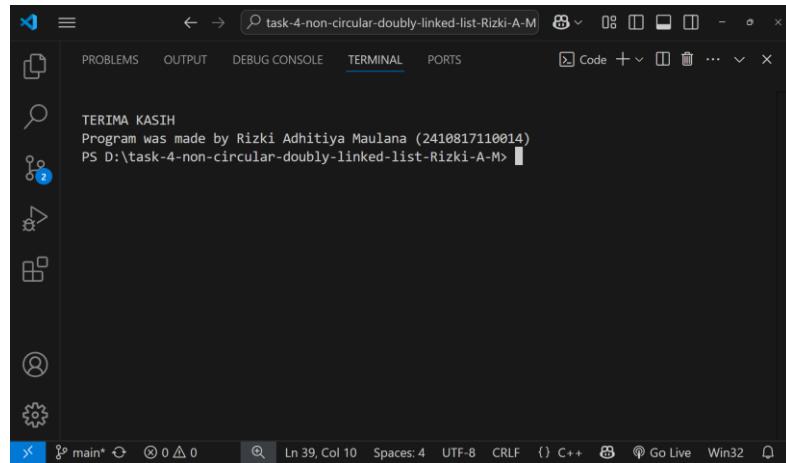
press any key to continue
```

Gambar 77. Reset Data Yang Ada



```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
3
Tidak terdapat data pada Linked List
press any key to continue
```

Gambar 78. Tampilan Data Setelah Dilakukan Reset Data



Gambar 79. Tampilan Keluar dari Program

C. Pembahasan

Pada baris [1] sampai [3] terdapat `#include` yang mana digunakan untuk mengakses sebuah file yang diinginkan. `<iostream>` yang ada digunakan untuk input dan output. Kemudian `<conio.h>` digunakan agar menyediakan fungsi fungsi yang berguna ketika ada interaksi langsung dengan keyboard, tanpa perlu menekan Enter. Terus `<stdlib.h>` digunakan untuk fungsi fungsi manajemen memori, konversi angka, kontrol proses, dan lingkungan program.

Pada baris [5] terdapat `using namespace std;` yang mana digunakan untuk menghindari penulisan std.

Pada baris [7] terdapat [11] terdapat `struct TNode` yang mana digunakan untuk menyimpan elemen-elemen dari linked list, dimana variabel `string data` digunakan untuk menyimpan isi atau informasi dari node tersebut seperti angka yang di input ke dalam list. Terus `Tnode *next` digunakan untuk menunjuk ke node berikutnya dalam urutan linked list, node terakhir yang ada dalam urutan linked list akan menunjuk kembali ke node pertama. Kemudian, `Tnode *prev` digunakan untuk menunjuk ke node sebelumnya dalam urutan linked list, sehingga memungkinkan dilakukannya penelusuran dua arah, baik maju (menggunakan next) maupun mundur (menggunakan prev).

Pada baris [13] terdapat `Tnode *head, *tail` yang mana `TNode *head` menunjuk pointer node pertama dan `TNode *tail` menunjuk pointer node terakhir.

Pada baris [15] sampai [17] terdapat `int pil dan menu` yang mana digunakan untuk menyimpan variabel Integer atau bilangan bulat. Terus `char pilihan [1]` yang mana digunakan untuk menyimpan variabel character, ditambah array sebagai batasan input dari user, `string dataBaru` yang digunakan untuk menyimpan variabel string atau katakter.

Pada baris [19] sampai [35] terdapat penamaan fungsi yang akan dimasukkan ke dalam program Linked list.

Pada baris [37] sampai [141] terdapat `int main()` yang mana digunakan untuk menjalankan dan menampilkan menu SLLC. Menu yang akan ditampilkan di dalam sistem ada sebanyak 3 buah, terdiri dari menu pertama untuk DLLNC dengan head, menu kedua DLLNC dengan head dan tail, menu terakhir untuk keluar. Setiap pilihan yang ada akan menampilkan tampilan berbeda sesuai dengan fungsi yang ada di dalam `switch case` yang dimasukkan pada program yang akan dijalankan. Terdapat `getch()` untuk menunggu tombol yang ditekan oleh pengguna dan membersihkan layar menggunakan `system("cls")`. Terus program akan terus berjalan selama user tidak memilih pilihan tiga (3) untuk keluar atau menghentikan program yang ada yang terletak di menu pilihan paling awal.

Pada baris [143] sampai [145] terdapat `void initH()` yang mana digunakan untuk menginisialisasikan kondisi awal dari linked list. `InitH()` akan berfokus dalam mengatur pointer atau variabel head ke dalam keadaan `NULL`.

Pada baris [147] sampai [150] terdapat `void initHT()` yang mana digunakan untuk menginisialisasikan kondisi awal dari linked list. `initHT()` akan berfokus dalam mengatur pointer atau variabel head ke dalam keadaan `NULL`, begitu juga untuk pointer atau variabel tail yang akan disetting atau diatur kedalam keadaan `NULL` seperti head.

Pada baris [152] sampai [155] terdapat `isEmptyH()` yang mana digunakan untuk melakukan pengecekan pada linked list, apakah head dalam keadaan kosong atau tidak. Fungsi ini akan mengembalikan nilai 1 apabila linked list dalam keadaan kosong dan 0 apabila tidak dalam keadaan kosong.

Pada baris [157] sampai [160] terdapat `isEmptyHT()` yang mana digunakan untuk melakukan pengecekan pada linked list, apakah head atau tail dalam keadaan

kosong atau tidak. Fungsi ini akan mengembalikan nilai 1 apabila linked list dalam keadaan kosong dan 0 apabila tidak dalam keadaan kosong.

Pada baris [162] sampai [178] terdapat `void tambahDepanH()` yang mana digunakan untuk menambahkan node baru ke bagian depan dari linked list, dimana pointer head yang menjadi patokannya. Apabila linked list dalam keadaan kosong, node yang baru ditambahkan akan menjadi head atau node baru. Kemudian, apabila linked list dalam keadaan tidak kosong, node yang baru ditambahkan akan menjadi head atau node pertama dan node yang sudah ada sebelumnya akan menjadi node kedua atau seterusnya.

Pada baris [180] sampai [197] terdapat `void tambahDepanHT()` yang mana digunakan untuk menambahkan node baru ke bagian depan dari linked list, dimana menggunakan head dan tail. Apabila linked list dalam keadaan kosong, head dan tail yang ada akan menunjuk ke node baru. Kemudian, apabila linked list dalam keadaan tidak kosong, node yang baru ditambahkan akan menjadi head atau node pertama dan node yang sudah ada sebelumnya akan menjadi node kedua atau seterusnya.

Pada baris [199] sampai [218] terdapat `void tambahBelakangH()` yang mana digunakan untuk menambahkan node baru ke bagian belakang dari linked list, tanpa bantuan tail yang membuat diperlukannya traversal dari head ke node terakhir, sehingga berkurangan efisiensi yang ada. Apabila linked list dalam keadaan kosong, node yang baru ditambahkan akan menjadi tail atau node terakhir. Kemudian, apabila linked list dalam keadaan tidak kosong, node yang baru ditambahkan akan menjadi tail atau node terakhir dan node yang sudah ada sebelumnya akan menjadi node sebelum node akhir.

Pada baris [220] sampai [237] terdapat `void tambahBelakangHT()` yang mana digunakan untuk menambahkan node baru ke bagian belakang dari linked list, dibantu dengan tail untuk menambahkan node dibagian terakhir, efisiensi yang ada meningkat. Apabila linked list dalam keadaan kosong, node yang baru ditambahkan akan menjadi tail atau node terakhir. Kemudian, apabila linked list dalam keadaan tidak kosong, node yang baru ditambahkan akan menjadi tail atau node terakhir dan node yang sudah ada sebelumnya akan menjadi node sebelum node akhir.

Pada baris [239] sampai [249] terdapat `void tampilanH()` yang mana digunakan untuk menampilkan seluruh isi linked list yang ada, dimulai dari depan hingga ke belakang. Data atau node yang akan ditampilkan akan dilakukan secara traversal yaitu dari *HEAD* (node pertama) hingga *NULL* (node terakhir).

Pada baris [251] sampai [261] terdapat `void tampilanHT()` yang mana digunakan untuk menampilkan seluruh isi linked list yang ada, dimulai dari depan hingga ke belakang. Data atau node yang akan ditampilkan akan dilakukan secara traversal yaitu dari *HEAD* (node pertama) hingga *TAIL* (node terakhir).

Pada baris [263] sampai [278] terdapat `void hapusDepanH()` yang mana digunakan untuk menghapus node pertama yang terdapat pada linked list, dimana pointer head yang menjadi patokannya. Apabila terdapat satu node saja pada linked list, maka setelah dilakukan hapus depan linked list head akan diatur menjadi kosong atau *NULL* menggunakan *fungsi InitH()*. Kemudian, apabila terdapat lebih dari satu node yang terdapat pada linked list, node pertama yang dihapus akan digantikan dengan elemen yang ada di setelah atau node kedua sebagai head terbaru.

Pada baris [280] sampai [295] terdapat `void hapusDepanHT()` yang mana digunakan untuk menghapus node pertama yang terdapat pada linked list, dimana menggunakan head dan tail. Apabila terdapat satu node saja pada linked list, maka setelah dilakukan hapus depan linked list head akan diatur menjadi kosong atau *NULL* menggunakan *fungsi InitHT()*. Kemudian, apabila terdapat lebih dari satu node yang terdapat pada linked list, node pertama yang dihapus akan digantikan dengan elemen yang ada di setelah atau node kedua sebagai head terbaru.

Pada baris [297] sampai [314] terdapat `void hapusBelakangH()` yang mana digunakan untuk menghapus node terakhir yang terdapat pada linked list, tanpa bantuan tail yang membuat diperlukannya traversal dari head ke node terakhir. Apabila terdapat satu node saja pada linked list, maka setelah dilakukan hapus belakang linked list akan diatur menjadi kosong atau *NULL* menggunakan *fungsi InitH()*. Kemudian, apabila terdapat lebih dari satu node yang terdapat pada linked list, node terakhir yang dihapus akan digantikan dengan elemen yang ada di sebelumnya atau node sebelum node akhir sebagai tail terbaru.

Pada baris [316] sampai [331] terdapat `void hapusBelakangHT()` yang mana digunakan untuk menghapus node terakhir yang terdapat pada linked list, dibantu dengan tail untuk menghapus node dibagian terakhir,. Apabila terdapat satu node saja pada linked list, maka setelah dilakukan hapus belakang linked list akan diatur menjadi kosong atau `NULL` menggunakan *fungsi InitHT()*. Kemudian, apabila terdapat lebih dari satu node yang terdapat pada linked list, node terakhir yang dihapus akan digantikan dengan elemen yang ada di sebelumnya atau node sebelum node akhir sebagai tail terbaru.

Pada baris [333] sampai [343] terdapat `void clearH()` yang mana digunakan untuk menghapus semua node yang ada pada linked list, baik dari node pertama hingga node terakhir. Setelah di lakukan penghapusan untuk semua node yang ada di dalam linked list, kemudian akan dipanggil *fungsi InitHT()* untuk mengatur linked list ke dalam kondisi kosong.

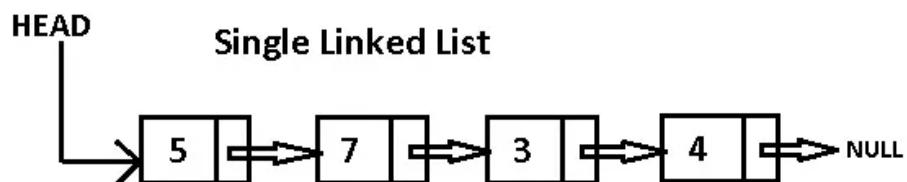
Pada baris [345] sampai [355] terdapat `void clearHT()` yang mana digunakan untuk menghapus semua node yang ada pada linked list, baik dari node pertama hingga node terakhir. Setelah di lakukan penghapusan untuk semua node yang ada di dalam linked list, kemudian akan dipanggil *fungsi InitHT()* untuk mengatur linked list ke dalam kondisi kosong.

SOAL 2

Apa fungsi next pada coding?

A. Pembahasan

Fungsi next dalam struktur data linked list dan double linked list digunakan untuk menunjuk ke node berikutnya dalam urutan data. Pada single linked list, setiap node mempunyai satu pointer saja yaitu next yang akan menunjuk ke node berikutnya. Dengan demikian, penelusuran dapat dilakukan dari node pertama (head) ke node terakhir dalam satu arah saja. Pada double linked list, next tetap berfungsi untuk menunjuk ke node berikutnya, tetapi pada struktur ini terdapat penunjuk tambahan yaitu prev yang memungkinkan untuk melakukan penelusuran dua arah.



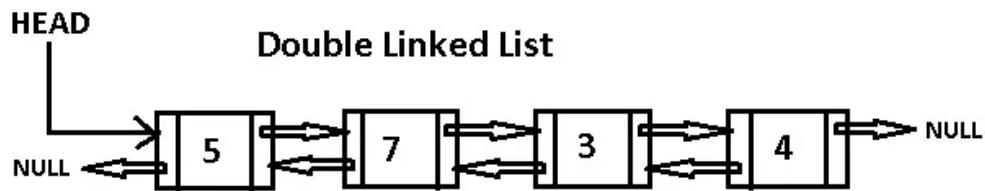
Gambar 80. Fungsi Next Di Single Linked List

SOAL 3

Apa fungsi prev pada coding?

A. Pembahasan

Fungsi prev hanya terdapat pada struktur double linked list, yang mana berfungsi untuk menunjuk ke node sebelumnya dari node yang sedang diakses. Dengan menggunakan prev, penjelajahan tidak hanya dapat dilakukan dengan satu arah yaitu maju dengan menggunakan next, tetapi juga dapat dilakukan dengan arah mundur dari node akhir ke node awal. Hal ini akan sangat berguna ketika dalam keadaan dimana kita harus bergerak mundur, menghapus elemen dari belakang, atau membalikkan arah penelusuran. Tanpa menggunakan prev, seperti pada single linked list, operasi-operasi yang ada akan menjadi lebih sulit dan kurang efisien karena kita harus memulai dari awal list setiap kali melakukan penelusuran.



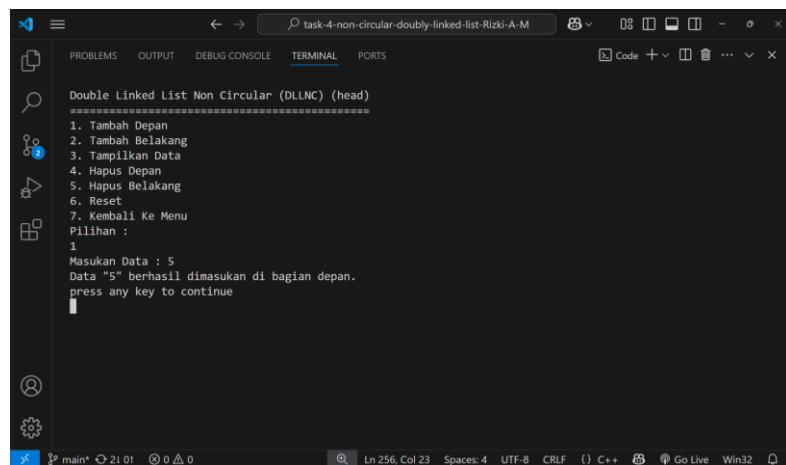
Gambar 81. Fungsi Prev Di Double Linked List

SOAL 4

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

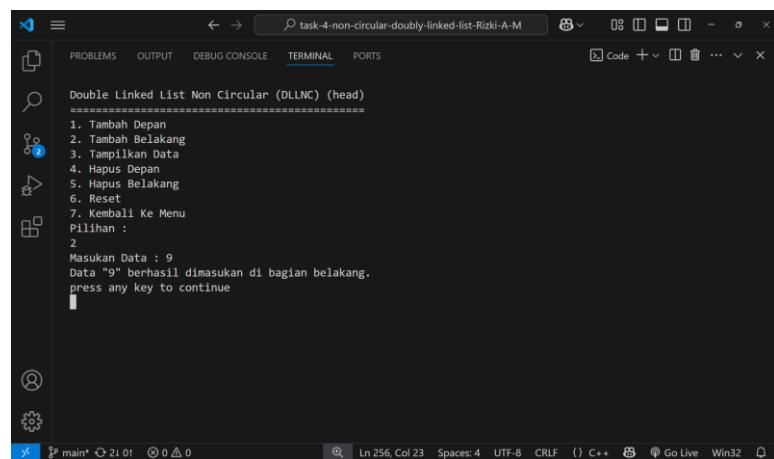
- Apa yang terjadi jika anda menambahkan beberapa data pada program lalu tampilkan datanya, dan screenshoot hasilnya.
- Jelaskan mengapa hal tersebut bisa terjadi dan data apa yang ditampilkan oleh program?

A. Output Program



```
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 :
1
Masukan Data : 5
Data "5" berhasil dimasukan di bagian depan.
press any key to continue
```

Gambar 82. Tampilan Tambah Depan Head



```
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 :
2
Masukan Data : 9
Data "9" berhasil dimasukan di bagian belakang.
press any key to continue
```

Gambar 83. Tampilan Tambah Belakang Head

```
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 :
5 5

press any key to continue
```

Gambar 84. Tampilan Data Head

```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
1

Masukan Data : 4
Data "4" berhasil dimasukan di bagian depan.

press any key to continue
```

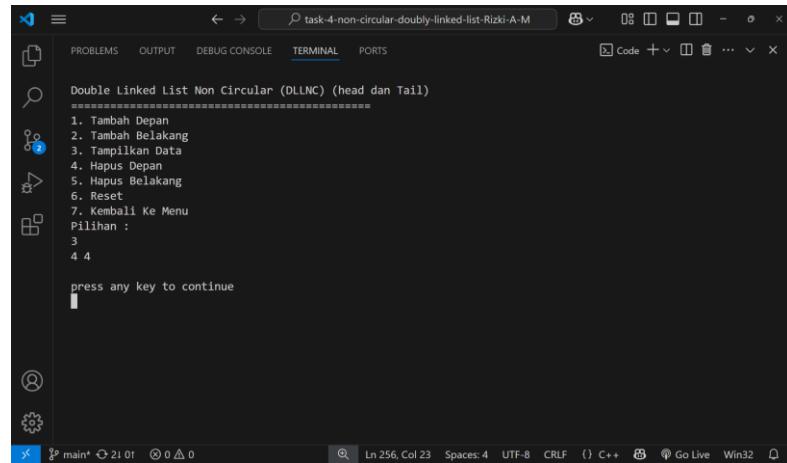
Gambar 85. Tampilan Tambah Depan Head Dan Tail

```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali Ke Menu
Pilihan :
2

Masukan Data : 8
Data "8" berhasil dimasukan di bagian belakang.

press any key to continue
```

Gambar 86. Tampilan Tambah Belakang Head Dan Tail



```
Double Linked List Non Circular (DLLNC) (head dan Tail)
=====
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

press any key to continue
```

Gambar 87. Tampilan Data Head Dan Tail

B. Pembahasan

Setelah dilakukan perubahan atau pergantian kode dari `cout<<bantu->data<< ' ';` menjadi `cout<<head->data<< ' ';`. Hal yang terjadi adalah program hanya akan mencetak data dari node pertama atau head yang terdapat dari linked list secara berulang-ulang mengikuti banyaknya data atau node yang ada di dalam linked list. Output yang keluar atau ditampilkan tidak menampilkan semua node yang dimasukkan sebelumnya dalam urutan, namun hanya mengulang data atau node pertama dalam output yang keluar. Hal ini berlaku baik di “DLLNC menggunakan head” dan “DLLNC menggunakan head dan tail” dari program yang ada.

Variabel bantu yang ada sebelumnya berguna untuk menunjukkan node pertama sampai node terakhir di dalam list. Namun karena variabel bantu yang ada diganti menjadi head, program hanya akan menunjukkan node pertama dan tidak menunjukkan node selanjutnya seperti urutan masuk atau node yang ada hingga tail.

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 88. Soal 1 Modul 5

A. Source Code

Tabel 8. Source Code Soal 1 Modul 5

```
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 string name, id;
11
12 void timeSort(const function<void()>& sortFunc,
13 const string& sortName)
14 {
15     auto start = chrono::high_resolution_clock::now();
16     sortFunc();
17     auto end = chrono::high_resolution_clock::now();
18     chrono::duration<double> duration = end - start;
19
20     cout << fixed << setprecision(10);
21     cout << sortName << " took " <<
22     duration.count() << " seconds\n";
23 }
24
25 void insertionSort(string &str)
26 {
27     for (int i = 1; i < str.size(); i++)
28     {
```

```

27         char key = str[i];
28         int j = i - 1;
29
30         while (j >= 0 && str[j] > key)
31         {
32             str[j + 1] = str[j];
33             j--;
34         }
35
36         str[j + 1] = key;
37     }
38 }
39
40 void merge(string &str, int left, int mid, int
right)
41 {
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     {
55         if (tempL[i] <= tempR[j])

```

```

56    {
57        str[k] = tempL[i];
58        i++;
59    }
60    else
61    {
62        str[k] = tempR[j];
63        j++;
64    }
65    k++;
66}
67
68 while (i < n1)
69 {
70    str[k] = tempL[i];
71    i++;
72    k++;
73}
74
75 while (j < n2)
76 {
77    str[k] = tempR[j];
78    j++;
79    k++;
80}
81
82 delete[] tempL;
83 delete[] tempR;
84}
85
86 void mergeSort(string &str, int left, int right)
87 {

```

```

88     if (left < right)
89     {
90         int mid = left + (right - left) / 2;
91         mergeSort(str, left, mid);
92         mergeSort(str, mid + 1, right);
93         merge(str, left, mid, right);
94     }
95 }
96
97 void shellSort(string &str, int n)
98 {
99     for (int gap = n/2; gap > 0; gap /= 2)
100    {
101        for (int i = gap; i < n; i++)
102        {
103            int temp = str[i];
104
105            int j;
106            for (j = i; j >= gap && str[j - gap]
107 > temp; j -= gap) str[j] = str[j - gap];
108            str[j] = temp;
109        }
110    }
111 }
112
113 void bubbleSort(string &str)
114 {
115     for (int i = 0; i < str.size() - 1; i++)
116     {
117         bool swapped = false;
118

```

```

119         for (int j = 0; j < str.size() - i - 1;
120             j++)
121             {
122                 if (str[j] > str[j + 1])
123                     {
124                         swap(str[j], str[j + 1]);
125                         swapped = true;
126                     }
127             }
128             if (!swapped) break;
129         }
130     }
131
132     int partition(string &str, int low, int high)
133     {
134         int pivot = str[high];
135         int i = (low - 1);
136
137         for (int j = low; j <= high - 1; j++)
138         {
139             if (str[j] <= pivot)
140             {
141                 i++;
142                 swap(str[i], str[j]);
143             }
144         }
145
146         swap(str[i + 1], str[high]);
147
148         return (i + 1);
149     }

```

```

150
151 void quickSort(string &str, int low, int high)
152 {
153     if (low < high)
154     {
155         int p_idx = partition(str, low, high);
156         quickSort(str, low, p_idx - 1);
157         quickSort(str, p_idx + 1, high);
158     }
159 }
160
161 void selectionSort(string &str)
162 {
163     for (int i = 0; i < str.size() - 1; i++)
164     {
165         int minIndex = i;
166
167         for (int j = i + 1; j < str.size(); j++)
168         {
169             if (str[j] < str[minIndex])
170             {
171                 minIndex = j;
172             }
173         }
174
175         swap(str[i], str[minIndex]);
176     }
177 }
178
179 int main()
180 {
181     int ch;

```

```

182     string temp;
183
184     do
185     {
186         cout << "=====+" << endl;
187         cout << "|      Sorting      |" << endl;
188         cout << "=====+" << endl;
189         cout << "| 1. Insertion Sort |" << endl;
190         cout << "| 2. Merge Sort   |" << endl;
191         cout << "| 3. Shell Sort   |" << endl;
192         cout << "| 4. Bubble Sort  |" << endl;
193         cout << "| 5. Quick Sort   |" << endl;
194         cout << "| 6. Selection Sort|" << endl;
195         cout << "| 7. Exit          |" << endl;
196         cout << "=====+" << endl;
197         cout << "Masukkan Pilihan: "; cin >> ch;
198
199         system("cls");
200
201         switch(ch)
202         {
203             case 1:
204                 cout << "Masukkan Nama: ";
205                 cin.ignore();
206                 getline(cin, name);
207
208                 system("cls");
209
210                 temp = name;
211                 cout << "Data Sebelum Diurutkan:
" << temp << endl;

```

```

212             timeSort([&]()
213             {insertionSort(temp); }, "Insertion Sort");
214             cout << "Data Setelah Diurutkan:
215             " << temp << endl;
216             break;
217             case 2:
218             cout << "Masukkan Nama: ";
219             cin.ignore();
220             getline(cin, name);
221             system("cls");
222             temp = name;
223             cout << "Data Sebelum Diurutkan:
224             " << temp << endl;
225             timeSort([&]() {mergeSort(temp,
226             0, temp.size() - 1); }, "Merge Sort");
227             cout << "Data Setelah Diurutkan:
228             " << temp << endl;
229             break;
230             case 3:
231             cout << "Masukkan Nama: ";
232             cin.ignore();
233             getline(cin, name);
234             system("cls");
235             temp = name;
236             cout << "Data Sebelum Diurutkan:
237             " << temp << endl;
238             timeSort([&]() {shellSort(temp,
239             temp.size()); }, "Shell Sort");

```

```

237             cout << "Data Setelah Diurutkan:
238             " << temp << endl;
239             break;
240             case 4:
241             cout << "Masukkan ID: "; cin >>
242             id;
243             system("cls");
244             temp = id;
245             cout << "Data Sebelum Diurutkan:
246             " << temp << endl;
247             timeSort([&] ()
248             {bubbleSort(temp); }, "Bubble Sort");
249             cout << "Data Setelah Diurutkan:
250             " << temp << endl;
251             break;
252             case 5:
253             cout << "Masukkan ID: "; cin >>
254             id;
255             system("cls");
256             temp = id;
257             cout << "Data Sebelum Diurutkan:
258             " << temp << endl;
259             timeSort([&] () {quickSort(temp,
260             0, temp.size() - 1); }, "Quick Sort");
261             cout << "Data Setelah Diurutkan:
262             " << temp << endl;
263             break;
264             case 6:

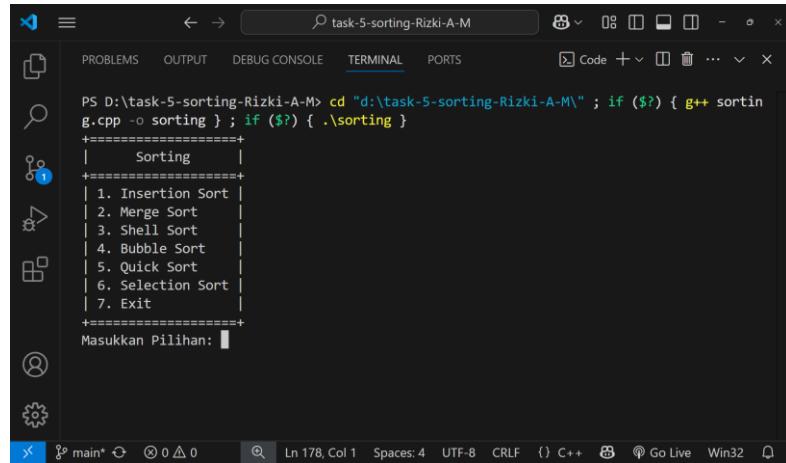
```

```

260         cout << "Masukkan ID: "; cin >>
261         id;
262         system("cls");
263
264         temp = id;
265         cout << "Data Sebelum Diurutkan:
266             " << temp << endl;
267             timeSort([&]()
268             {selectionSort(temp); }, "Selection Sort");
269             cout << "Data Setelah Diurutkan:
270             " << temp << endl;
271             break;
272         case 7:
273             cout << "Terima Kasih" << endl;
274             cout << "This Program Was Made by
275             Rizki Adhitiya Maulana (2410817110014)" << endl;
276             break;
277         default:
278             cout << "Opsi Tidak Valid.
279             Silahkan Coba Lagi." << endl;
280             }
281             cout << "Press any key to continue...";
282             getch();
283             system("cls");
284         }
285         while (ch != 7);
286
287         return 0;
288     }

```

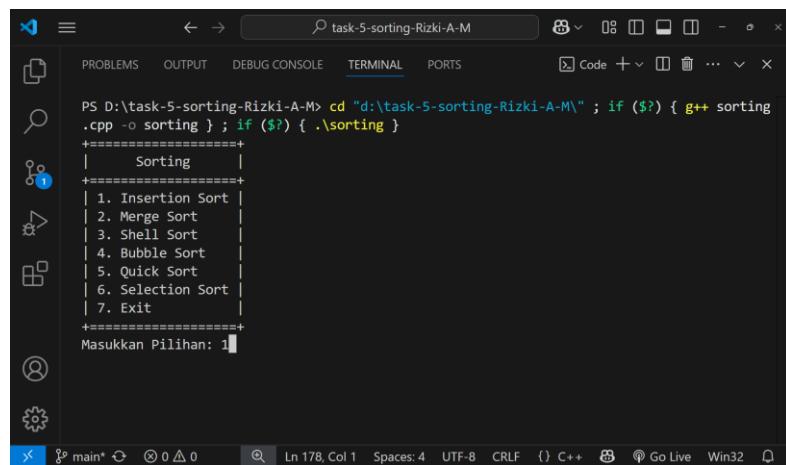
B. Output Program



```
PS D:\task-5-sorting-Rizki-A-M> cd "d:\task-5-sorting-Rizki-A-M\"; if ($?) { g++ sorting
.g.cpp -o sorting } ; if ($?) { .\sorting }

+=====+
|   Sorting   |
+=====+
| 1. Insertion Sort |
| 2. Merge Sort    |
| 3. Shell Sort    |
| 4. Bubble Sort   |
| 5. Quick Sort    |
| 6. Selection Sort|
| 7. Exit          |
+=====+
Masukkan Pilihan: 1
```

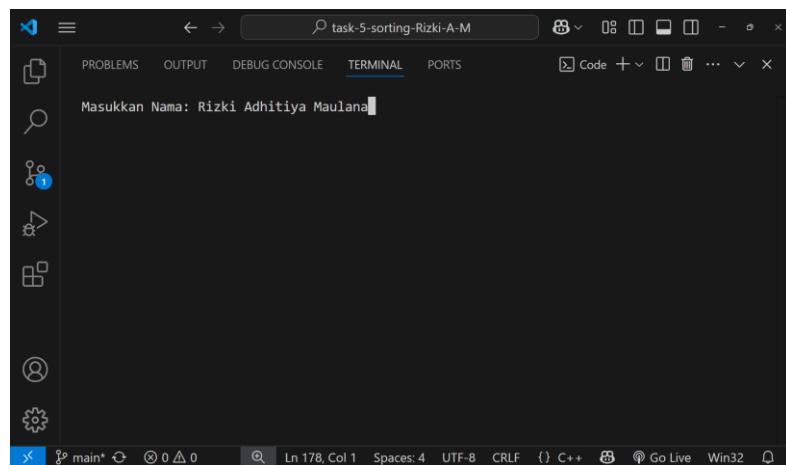
Gambar 89. Tampilan Menu Sorting



```
PS D:\task-5-sorting-Rizki-A-M> cd "d:\task-5-sorting-Rizki-A-M\"; if ($?) { g++ sorting
.g.cpp -o sorting } ; if ($?) { .\sorting }

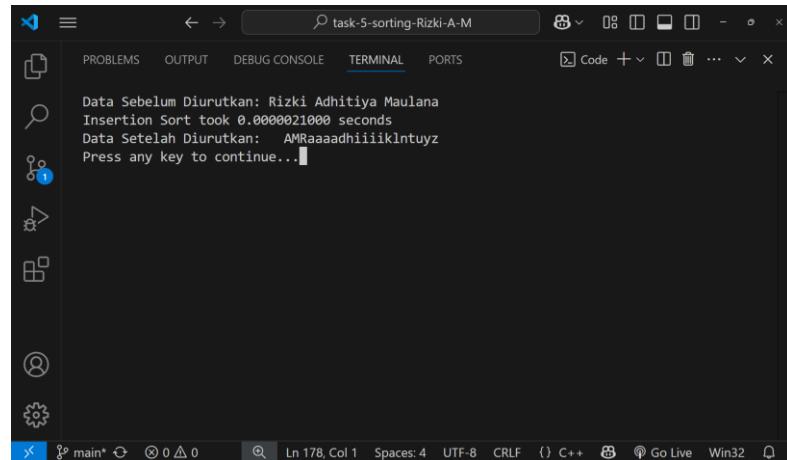
+=====+
|   Sorting   |
+=====+
| 1. Insertion Sort |
| 2. Merge Sort    |
| 3. Shell Sort    |
| 4. Bubble Sort   |
| 5. Quick Sort    |
| 6. Selection Sort|
| 7. Exit          |
+=====+
Masukkan Pilihan: 1
```

Gambar 90. Pilihan 1 Pada Menu Sorting



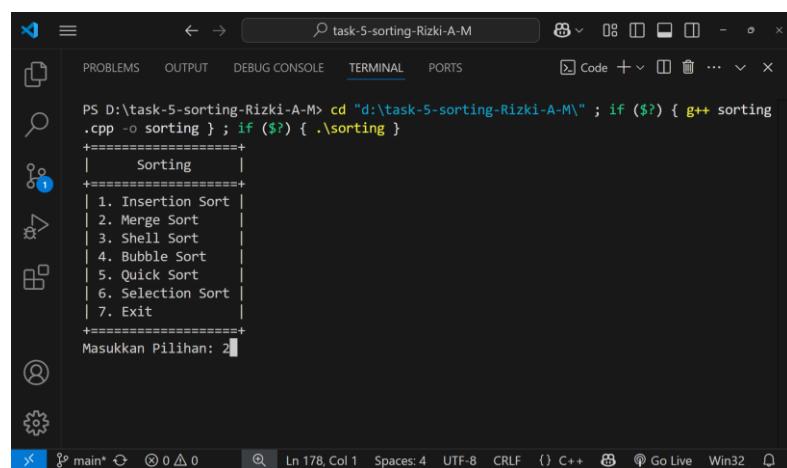
```
Masukkan Nama: Rizki Adhitya Maulana
```

Gambar 91. Masukkan Nama Pada Pilihan 1



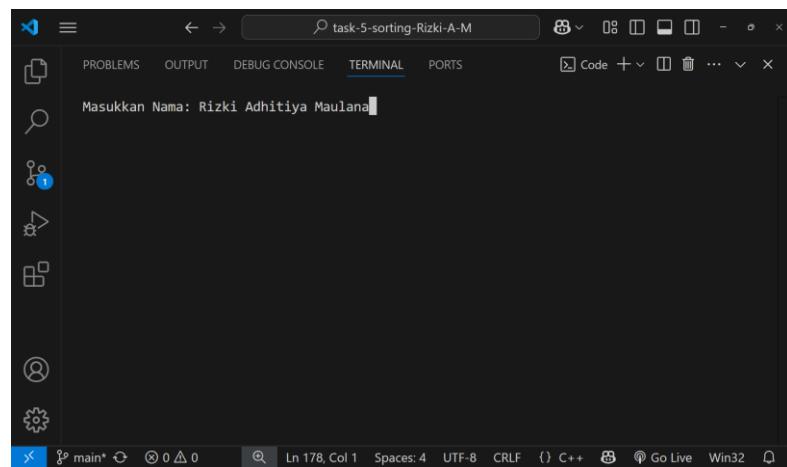
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Data Sebelum Diurutkan: Rizki Adhitiya Maulana
Insertion Sort took 0.0000021000 seconds
Data Setelah Diurutkan: AMRaaaadhiiiiklntuyz
Press any key to continue...
```

Gambar 92. Tampilan Hasil Dari Insertion Sort



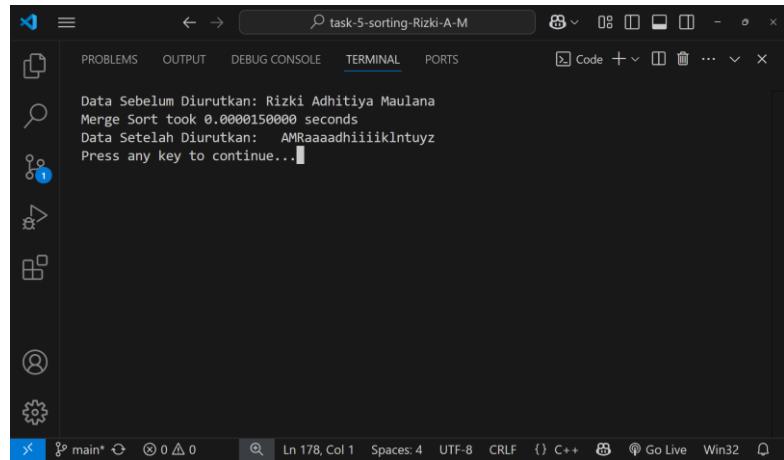
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\task-5-sorting-Rizki-A-M> cd "d:\task-5-sorting-Rizki-A-M\" ; if ($?) { g++ sorting.cpp -o sorting } ; if ($?) { .\sorting }
+=====+
|   Sorting   |
+=====+
| 1. Insertion Sort |
| 2. Merge Sort |
| 3. Shell Sort |
| 4. Bubble Sort |
| 5. Quick Sort |
| 6. Selection Sort |
| 7. Exit |
+=====+
Masukkan Pilihan: 2
```

Gambar 93. Pilihan 2 Pada Menu Sorting



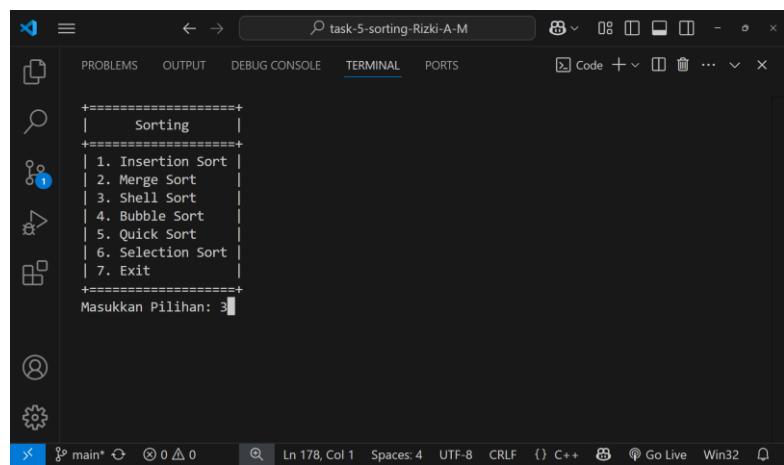
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukkan Nama: Rizki Adhitiya Maulana
```

Gambar 94. Masukkan Nama Pada Pilihan 2



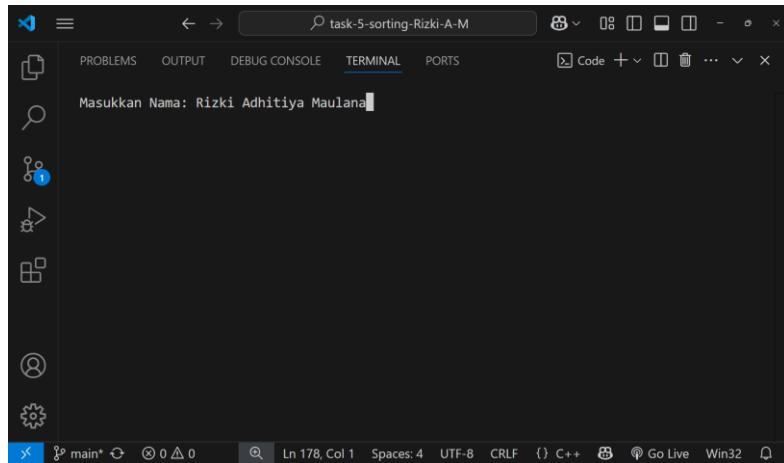
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Data Sebelum Diurutkan: Rizki Adhitiya Maulana
Merge Sort took 0.0000150000 seconds
Data Setelah Diurutkan: AMRaaadhiiiiklntuyz
Press any key to continue...
```

Gambar 95. Tampilan Hasil Dari Merge Sort



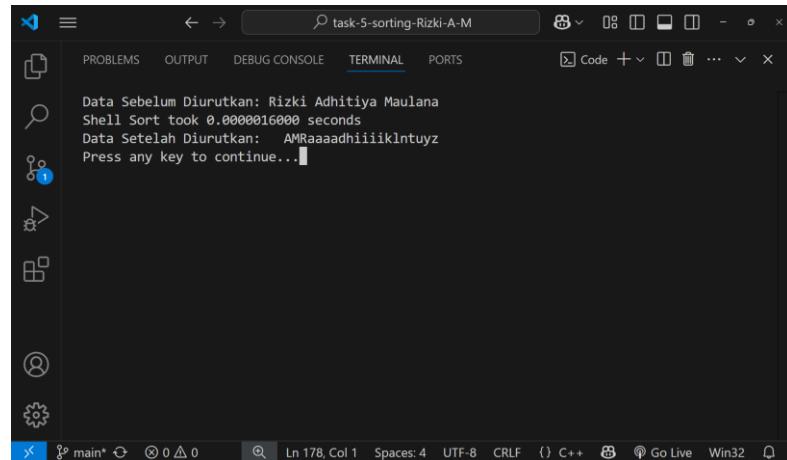
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+=====+
|   Sorting   |
+=====+
| 1. Insertion Sort |
| 2. Merge Sort |
| 3. Shell Sort |
| 4. Bubble Sort |
| 5. Quick Sort |
| 6. Selection Sort |
| 7. Exit |
+=====+
Masukkan Pilihan: 3
```

Gambar 96. Pilihan 3 Pada Menu Sorting



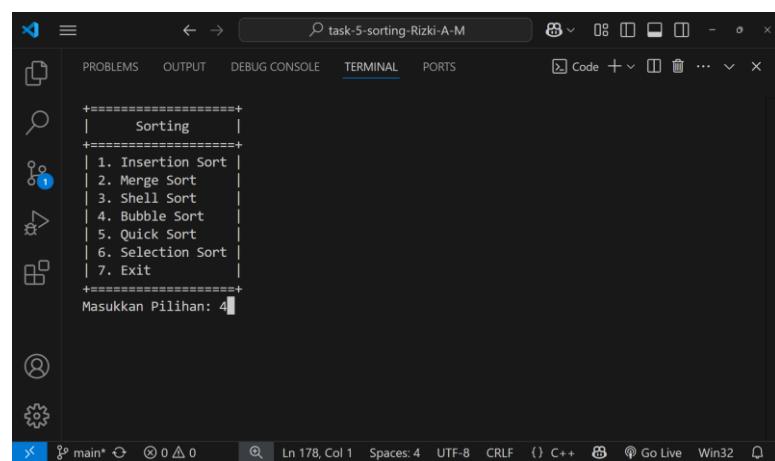
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukkan Nama: Rizki Adhitiya Maulana
```

Gambar 97. Masukkan Nama Pada Pilihan 3



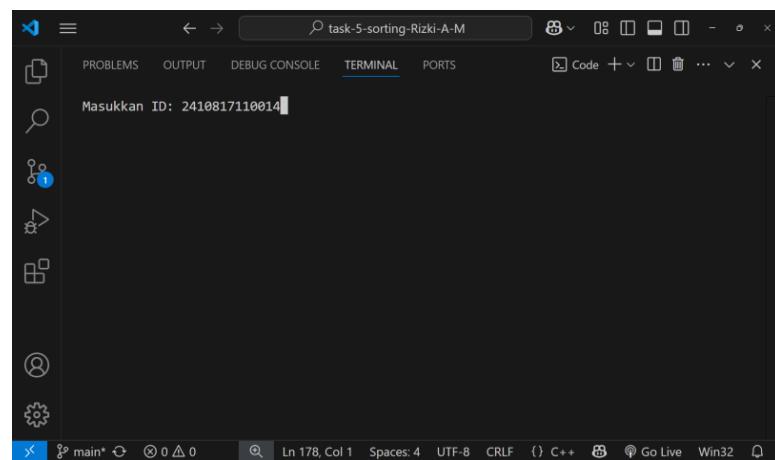
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Data Sebelum Diurutkan: Rizki Adhiyya Maulana
Shell Sort took 0.0000016000 seconds
Data Setelah Diurutkan: AMRaaaadhiiiiklntuyz
Press any key to continue...
```

Gambar 98. Tampilan Hasil Dari Shell Sort



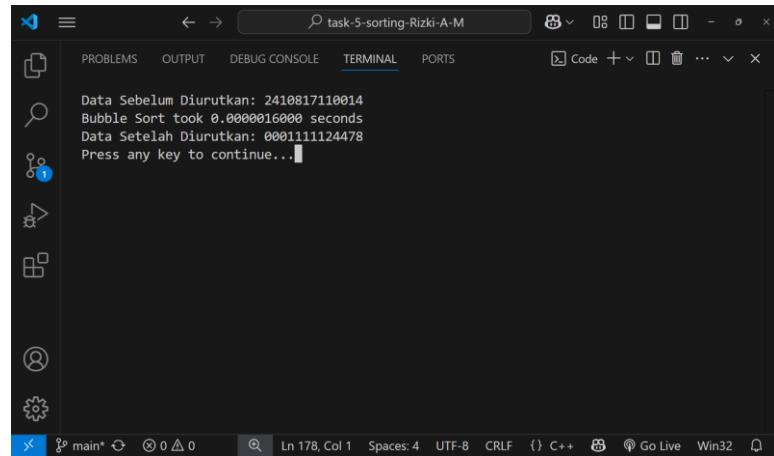
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+=====+
| Sorting |
+=====+
| 1. Insertion Sort |
| 2. Merge Sort |
| 3. Shell Sort |
| 4. Bubble Sort |
| 5. Quick Sort |
| 6. Selection Sort |
| 7. Exit |
+=====+
Masukkan Pilihan: 4
```

Gambar 99. Pilihan 4 Pada Menu Sorting



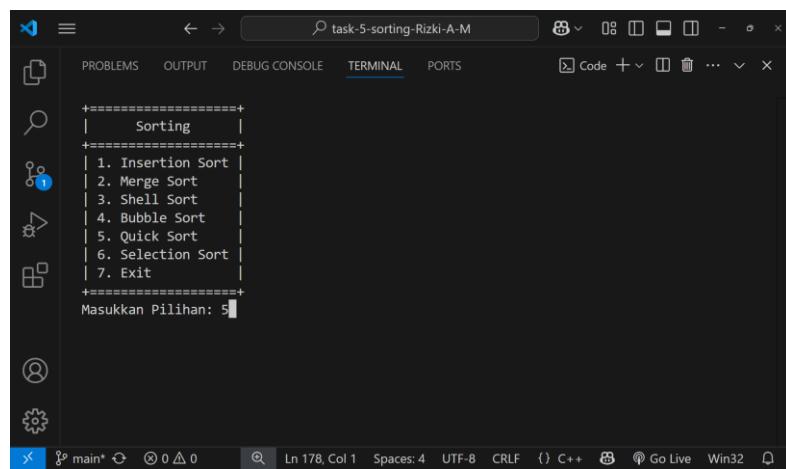
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukkan ID: 2410817110014
```

Gambar 100. Masukkan ID Pada Pilihan 4



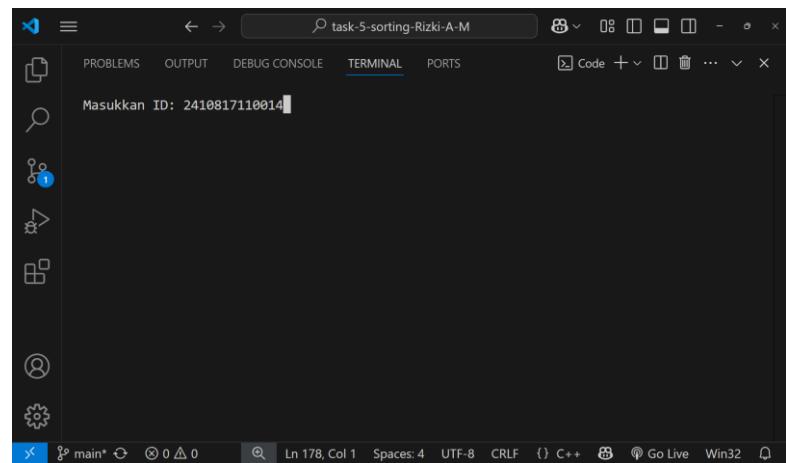
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Data Sebelum Diurutkan: 2410817110014
Bubble Sort took 0.0000016000 seconds
Data Setelah Diurutkan: 000111124478
Press any key to continue...
```

Gambar 101. Tampilan Hasil Dari Bubble Sort



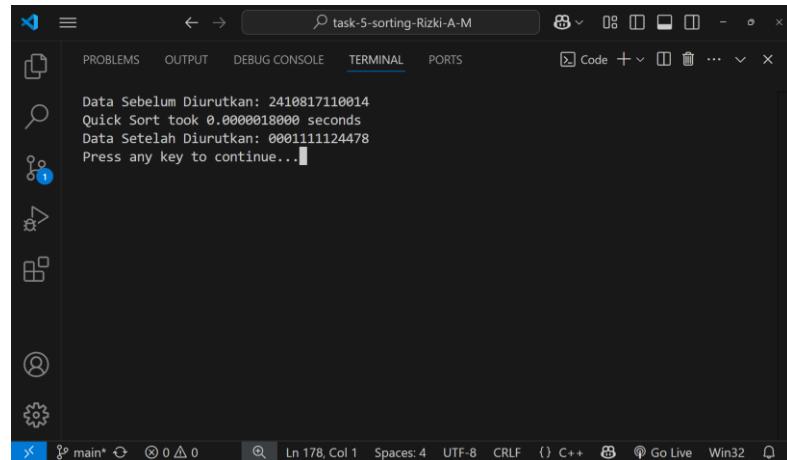
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
=====
|       Sorting      |
=====
| 1. Insertion Sort |
| 2. Merge Sort     |
| 3. Shell Sort     |
| 4. Bubble Sort    |
| 5. Quick Sort     |
| 6. Selection Sort|
| 7. Exit           |
=====
Masukkan Pilihan: 5
```

Gambar 102. Pilihan 5 Pada Menu Sorting



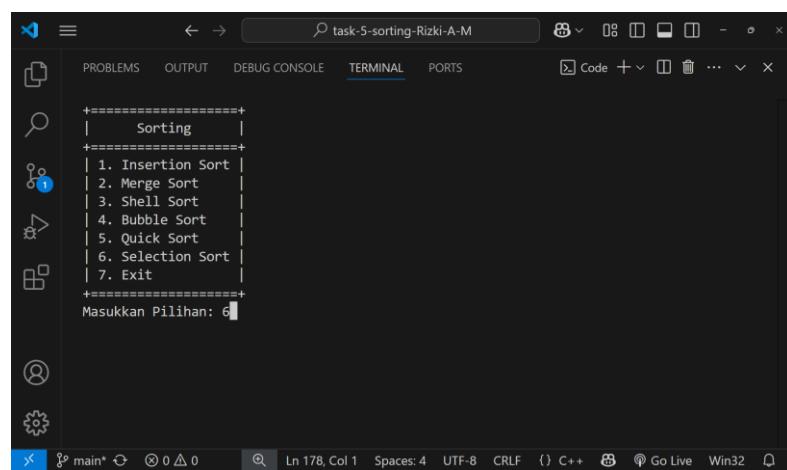
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukkan ID: 2410817110014
```

Gambar 103. Masukkan ID Pada Pilihan 5



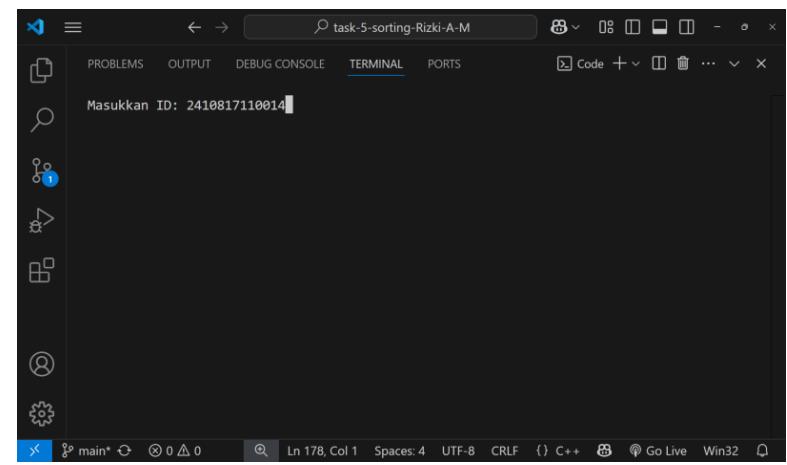
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Data Sebelum Diurutkan: 2410817110014
Quick Sort took 0.000018000 seconds
Data Setelah Diurutkan: 0001111124478
Press any key to continue...[
```

Gambar 104. Tampilan Hasil Dari Quick Sort



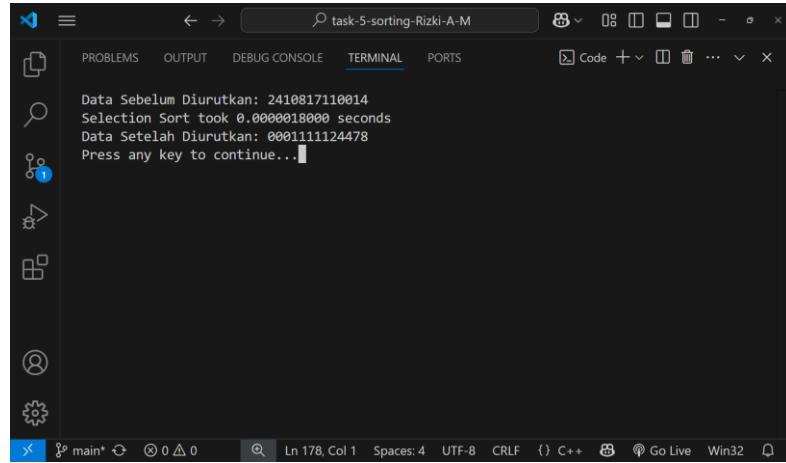
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+=====+
|      Sorting      |
+=====+
| 1. Insertion Sort |
| 2. Merge Sort     |
| 3. Shell Sort     |
| 4. Bubble Sort    |
| 5. Quick Sort      |
| 6. Selection Sort |
| 7. Exit           |
+=====+
Masukkan Pilihan: 6[
```

Gambar 105. Pilihan 6 Pada Menu Sorting



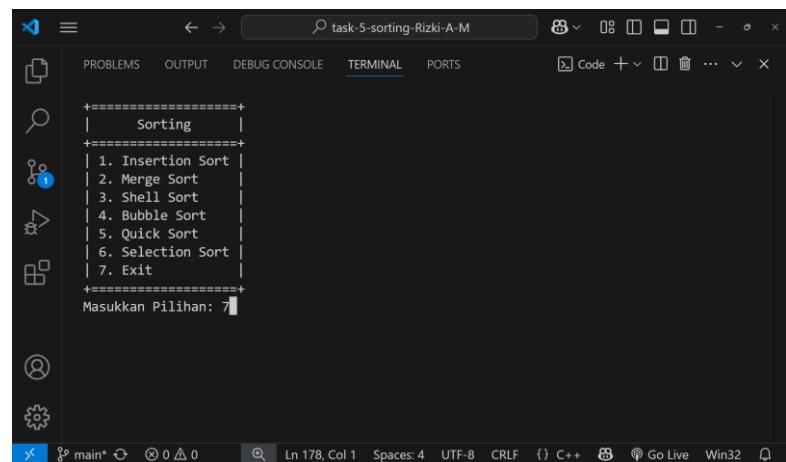
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukkan ID: 2410817110014[
```

Gambar 106. Masukkan ID Pada Pilihan 6



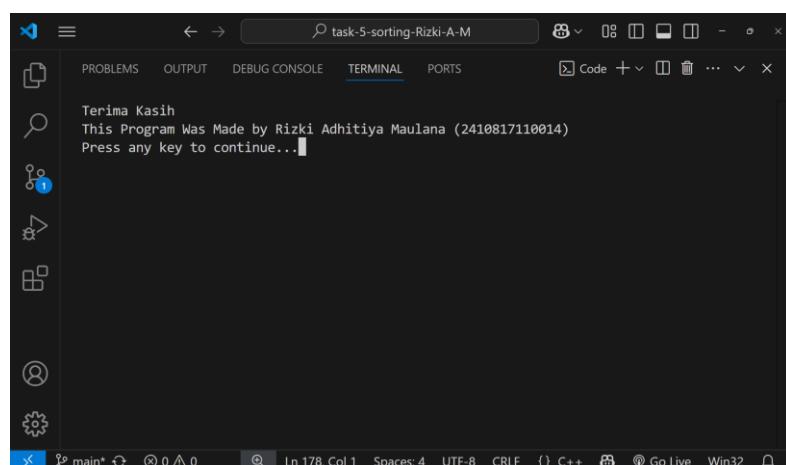
```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Data Sebelum Diurutkan: 2410817110014
Selection Sort took 0.0000018000 seconds
Data Setelah Diurutkan: 0001111124478
Press any key to continue...
```

Gambar 107. Tampilan Hasil Dari Selection Sort



```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+=====+
|   Sorting   |
+=====+
| 1. Insertion Sort |
| 2. Merge Sort |
| 3. Shell Sort |
| 4. Bubble Sort |
| 5. Quick Sort |
| 6. Selection Sort |
| 7. Exit         |
+=====+
Masukkan Pilihan: 7
```

Gambar 108. Pilihan 7 Pada Menu Sorting



```
task-5-sorting-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Terima Kasih
This Program Was Made by Rizki Adhitiya Maulana (2410817110014)
Press any key to continue...
```

Gambar 109. Tampilan Program Selesai

C. Pembahasan

- **Alur Program**

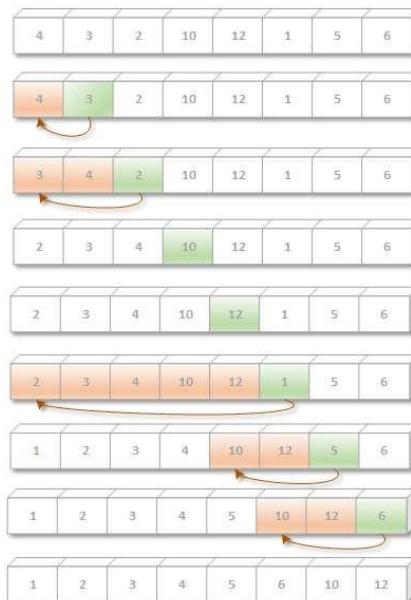
Ketika program dimulai, akan muncul tampilan menu dengan nama "*Sorting*" yang di dalamnya terdapat tujuh opsi, diantaranya ada opsi pertama untuk "*Insetion Sort*", opsi kedua "*Merge Sort*", opsi ketiga "*Shell Sort*", opsi keempat "*Bubble Sort*", opsi kelima "*Quick Sort*", opsi keenam "*Selection Sort*" dan opsi ketujuh "*Exit*". Program yang ada akan selalu berjalan atau melakukan perulangan karena adanya struktur "*do-while*", yang akan terus menampilkan menu sorting sampai pengguna memilih opsi "*Exit*". Kemudian, setiap selesai menentukan pilihan atau melakukan input untuk menu apa yang akan diakses, *fungsi* atau *perintah* "*CLS*" akan selalu dijalankan untuk membuat tampilan dari program terlihat tidak menumpuk dan rapi.

Struktur "*Switch-case*" digunakan untuk semua pilihan yang ada, mulai dari *case 1* sampai *case 3*, pengguna akan diminta untuk memasukkan atau mengimput "*nama*". Sedangkan, *case 4* sampai *case 6* pengguna akan diminta untuk memasukkan atau mengimput "*id*". Kecuali *case 7* yang akan menampilkan ucapan terima kasih dan nama serta nim dari pembuat program, dan *case default* yang akan memberitahukan bahwa opsi tidak valid karena pengguna memasukan pilihan di luar dari tujuh opsi yang ada. Setelah selesai melakukan input baik itu berupa "*nama*" atau "*nim*", program akan menampilkan penampakan string sebelum dan sesudah disortir. Pada *case 1*, program akan menjalankan *fungsi* "*Insertion Sort*" dan menampilkan durasi eksekusinya, *case 2* menjalankan *fungsi* "*Merge Sort*" beserta durasinya, *case 3* menjalankan *fungsi* "*Shell Sort*" dengan durasinya, *case 4* menjalankan *fungsi* "*Bubble Sort*" disertai durasi, *case 5* menjalankan *fungsi* "*Quick Sort*" dengan waktu eksekusinya, dan *case 6* menjalankan *fungsi* "*Selection Sort*" serta menampilkan durasinya.

Setelah fungsi dari setiap pilihan dari program menyelesaikan semua tugas dan memprintkan semua hasil prosesnya, program akan menunggu input tombol apa saja untuk melanjutkan, lalu membersihkan layar dan menampilkan tampilan menu “*Sorting*” kembali.

- **Insertion Sort**

Insertion Sort digunakan untuk melakukan sebuah pengurutan data, yang mana dengan cara membandingkan elemen yang ada pada array. Setiap elemen dalam array akan dibandingkan dengan elemen sebelumnya secara berurutan hingga semua elemen menemukan posisinya masing-masing. Proses pengurutannya di mulai dari elemen kedua, kemudian elemen kedua akan dibandingkan dengan elemen sebelumnya. Apabila elemen kedua lebih kecil dari elemen sebelumnya, maka elemen kedua akan digeser ke kiri atau disisipkan sebelum elemen yang lebih besar dari elemen kedua. Namun, apabila elemen kedua lebih besar dari elemen sebelumnya, maka tidak akan ada pergeseran posisi. Proses ini akan terus diulangi hingga elemen ketiga, keempat dan yang terakhir terurut.



Gambar 110. Ilustrasi Insertion Sort

Dalam pembahasan fungsinya, `void insertionSort(string &str)` merupakan implementasi Insertion Sort yang mengurutkan karakter-karakter dalam sebuah string secara langsung karena menggunakan `pass by reference`. Proses pengurutan dimulai dengan `loop for` dari indeks `i = 1`

(karakter kedua), di mana setiap karakter pada $str[i]$ disimpan sebagai key yang akan disisipkan ke bagian string di sebelah kirinya yang sudah terurut. Variabel j diinisialisasi dengan $i - 1$ untuk melacak posisi di bagian yang sudah terurut. Selanjutnya, loop `while` akan menggeser karakter $str[j]$ satu posisi ke kanan ($str[j + 1] = str[j]$) selama j masih valid ($j \geq 0$) dan $str[j]$ lebih besar dari key , sambil terus mengurangi j ($j--$) untuk mencari posisi yang tepat. Setelah loop `while` selesai, key ditempatkan pada posisi $str[j + 1]$, menyelesaikan proses penyisipan satu karakter dan secara bertahap mengurutkan seluruh string.

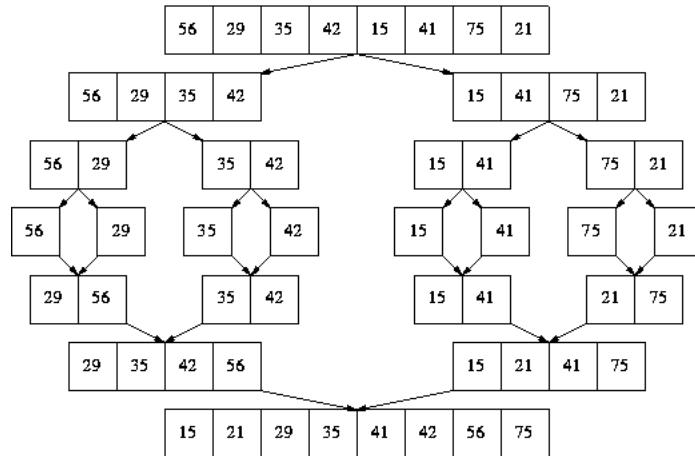
Kompleksitas waktu dari Insertion Sort tergantung pada kondisi awal data yang akan diurutkan. Pada kasus terbaik (best case) kompleksitas waktu Insertion Sort adalah $O(n)$, karena data yang ada sudah terurut, algoritma yang ada hanya akan melakukan perbandingan antar elemen atau pengecekan. Pada kasus rata-rata (average case) dan kasus terburuk (worst case) kompleksitas waktunya adalah $O(n^2)$, karena saat data terurut acak atau terbalik, algoritma yang ada harus membandingkan dan menggeser hampir semua elemen. Sementara itu, kompleksitas ruang dari Insertion Sort adalah $O(1)$ karena algoritma Insertion Sort tidak memerlukan struktur data tambahan.

Jadi kesimpulannya, Insertion Sort ini sangat cocok digunakan untuk mengurutkan data jumlah yang kecil atau hampir terurut. Kemudian juga hemat ruang penyimpanan dalam penggunaannya karena tidak perlu tambahan struktur data dalam prosesnya, hanya beberapa variabel bantu sebagai penyimpanan sementara.

- **Merge Sort**

Merge Sort digunakan untuk melakukan sebuah pengurutan data, yang mana menggunakan pendekatan divide and conquer dalam prosesnya. Cara kerja pengurutannya di mulai dengan membagi kumpulan data dalam satu array menjadi dua bagian secara terus menerus hingga menjadi bagian paling kecil yang berisi satu elemen. Setelah proses pembagian kumpulan data telah selesai, bagian paling kecil yang ada akan digabung kembali. Dalam proses penggabungannya akan selalu melibatkan dua bagian yang telah dibagi

sebelumnya, karena ketika digabungkan elemen yang ada di dua bagian tersebut akan dibandingkan dan diurutkan dari yang terkecil disebelah kiri hingga yang terbesar di sebelah kanan. Proses penggabungan akan terus dilakukan secara bertahap dengan memperhatikan urutan elemen agar tetap terurut hingga seluruh elemen yang dibagi sebelumnya kembali menjadi satu kesatuan.



Gambar 111. Ilustrasi Merge Sort

Dalam pembahasan fungsinya, `void mergeSort(string &str, int left, int right)` adalah bagian yang bekerja secara rekursif untuk mengurutkan string. Prosesnya diawali dengan kondisi dasar `if (left < right)`, yang akan terus membagi string menjadi bagian-bagian yang lebih kecil hingga hanya tersisa satu karakter (yang otomatis terurut), menghentikan rekursi. Kemudian, dua panggilan rekursif dilakukan: satu untuk mengurutkan paruh kiri (`mergeSort(str, left, mid)`) dan satu untuk mengurutkan paruh kanan (`mergeSort(str, mid + 1, right)`). Selanjutnya dilanjutkan dengan fungsi `void merge(string &str, int left, int mid, int right)` untuk menggabungkan dua sub array yang sudah terurut (yang direpresentasikan oleh bagian-bagian dari str dari left hingga mid, dan dari mid + 1 hingga right) menjadi satu bagian yang terurut dalam string asli. Fungsi ini pertama menghitung ukuran kedua sub-array (n1 dan n2), lalu mengalokasikan dua array sementara (`tempL` dan `tempR`) di heap untuk menyalin elemen-elemen dari sub-array yang bersangkutan. Proses penggabungan dilakukan dengan

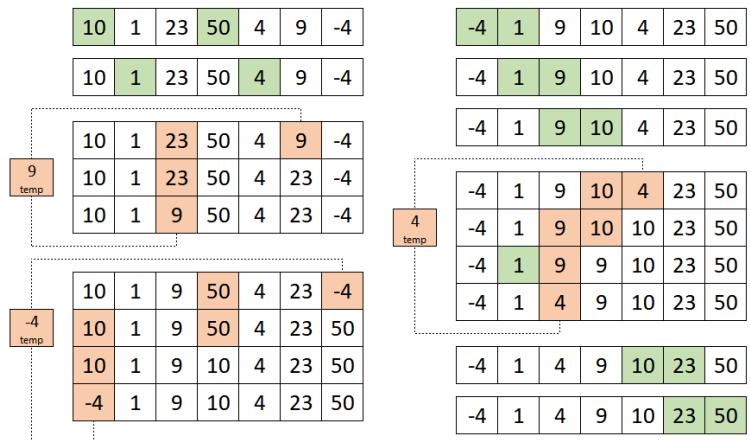
loop while utama yang membandingkan elemen-elemen dari $tempL[i]$ dan $tempR[j]$, menyalin karakter yang lebih kecil ke $str[k]$ dan menginkrementasi indeks yang relevan (i, j, dan k). Setelah salah satu array sementara habis, dua loop while terpisah akan menyalin sisa elemen dari array sementara yang belum habis ke str. Terakhir, memori yang dialokasikan secara dinamis dibebaskan menggunakan `delete[]` untuk mencegah memory leak, memastikan operasi penggabungan berjalan efisien dan bersih.

Kompleksitas waktu dari Merge Sort tidak tergantung pada kondisi awal data yang akan diurutkan. Artinya baik dalam kasus baik, kasus rata-rata, ataupun kasus yang buruk kompleksitas waktu Merge Sort adalah $O(n \log n)$, karena ada setiap tahap penggabungan, seluruh elemen (n elemen) harus diproses. Sementara itu, kompleksitas ruang dari Merge Sort adalah $O(n)$ karena algoritma Merge Sort memerlukan ruang tambahan sebagai tempat penyimpanan sementara selama proses penggabungan.

Jadi kesimpulannya, Merge Sort ini cocok digunakan untuk mengurutkan data dalam jumlah yang besar karena hasil dari pengurutan datanya yang akurat dan waktu eksekusi data yang konsisten di semua kasus, namun membutuhkan ruang tambahan untuk proses di dalamnya untuk menyimpan array sementara saat prosesnya.

- **Shell Sort**

Shell Sort merupakan pengembangan dari Insertion Sort, yang mana Shell Sort digunakan untuk melakukan sebuah pengurutan data dengan cara membandingkan dan menukar elemen-elemen yang memiliki jarak terlebih dahulu. Jarak antar elemen ini disebut dengan "gap". Cara kerja pengurutannya di mulai dengan membandingkan data yang letaknya jauh terlebih dahulu, lalu jarak yang ada tersebut dikurangi sedikit demi sedikit secara bertahap hingga 1. Apabila elemen gap yang ada berupa 1, maka algoritma yang ada akan melakukan Insertion Sort.



Gambar 112. Ilustrasi Shell Sort

Dalam pembahasan fungsinya, `void shellSort(string &str, int n)` mengimplementasikan algoritma Shell Sort, sebuah peningkatan dari Insertion Sort yang lebih efisien, terutama untuk data berukuran sedang. Algoritma ini bekerja dengan pendekatan "*gap*" atau jarak, dimulai dengan `loop for` terluar yang mengatur ukuran gap awal (biasanya $n/2$) dan secara bertahap mengurangi gap menjadi setengahnya hingga mencapai 1. Untuk setiap nilai gap, `loop for` tengah akan mengiterasi string dari posisi gap hingga n , mengambil setiap karakter `str[i]` sebagai `temp`. Kemudian, `loop for` terdalam yang mirip Insertion Sort akan membandingkan `temp` dengan elemen-elemen yang berada gap posisi di belakangnya (`str[j - gap]`). Jika `str[j - gap]` lebih besar dari `temp`, maka `str[j - gap]` digeser ke posisi `str[j]`, dan proses ini berlanjut mundur dengan mengurangi j sebesar gap hingga posisi yang tepat ditemukan. Akhirnya, `temp` disisipkan pada posisi `str[j]`, secara efektif mengurutkan sub-list yang berjarak gap dan secara bertahap mengurutkan seluruh string ketika gap mengelil hingga 1.

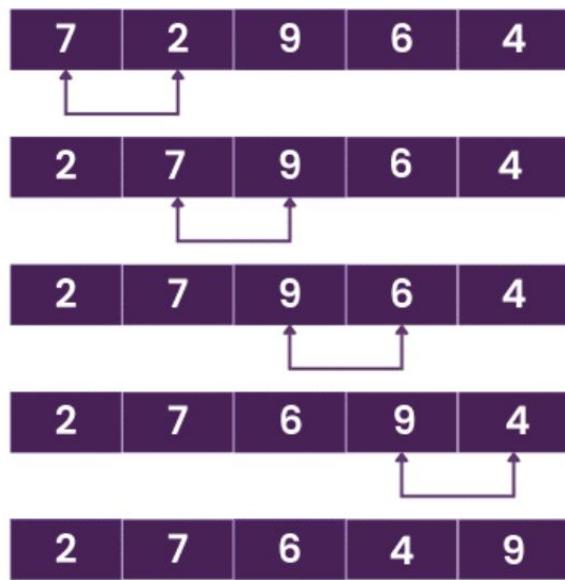
Kompleksitas waktu dari Shell Sort bergantung pada kondisi awal data yang akan diurutkan dan gap yang ada. Pada kasus terbaik (best case) kompleksitas waktu Shell Sort adalah $\Omega(n \log(n))$, karena data yang ada sudah terurut, algoritma yang ada hanya akan melakukan perbandingan antar elemen atau pengecekan. Pada kasus rata-rata (average case) kompleksitas waktunya adalah $O(n * \log n) \sim O(n^{1.25})$, karena data yang ada sudah

hampi terurut hanya perlu melakukan sedikit pergeseran diakhir dengan Insertion Sort. Kasus terburuk (worst case) kompleksitas waktunya adalah $O(n^2)$, karena saat data terurut acak atau terbalik, algoritma yang ada harus membandingkan dan memindahkan dan menggeser hampir semua elemen. Sementara itu, kompleksitas ruang dari Shell Sort adalah $O(1)$ karena algoritma Insertion Sort tidak memerlukan struktur data tambahan.

Jadi kesimpulan, Shell Sort ini cocok digunakan untuk mengurutkan data dalam jumlah yang sedang atau hampir terurut. Kemudian juga hemat ruang penyimpanan dalam penggunaannya karena tidak perlu tambahan struktur data dalam prosesnya, hanya beberapa variabel bantu sebagai penyimpanan sementara.

- **Bubble Sort**

Bubble Sort digunakan untuk melakukan sebuah pengurutan data, yang mana dengan mendorong elemen terbesar yang ada di dalam kumpulan data ke kanan secara bertahap. Cara kerja penggurutannya dimulai dengan membandingkan elemen pertama dan elemen kedua. Apabila elemen pertama lebih besar dari elemen kedua maka posisi dari keduanya akan di tukar, elemen yang lebih besar akan berpindah ke kanan. Kemudian, perbandingan berlanjut antara elemen kedua dan ketiga, dan begitu seterusnya hingga elemen terakhir. Setelah satu lintasan selesai, elemen terbesar akan berada di posisi paling kanan. Proses ini diulang kembali untuk elemen-elemen yang tersisa, dengan setiap lintasan mengurangi jumlah elemen yang perlu diperiksa karena elemen-elemen terbesar sudah berada di posisi yang benar. Pengurutan akan berhenti ketika dalam satu lintasan tidak ada lagi pertukaran, yang menandakan bahwa seluruh data sudah dalam keadaan terurut.



Gambar 113. Ilustrasi Bubble Sort

Dalam pembahasan fungsinya, `void bubbleSort(string &str)` bekerja melalui *dua loop for* bersarang, pada *loop terluar* (*i*) menentukan berapa banyak elemen terbesar yang sudah dipindahkan ke posisi akhirnya di bagian kanan string, sementara pada *loop dalam* (*j*) melakukan perbandingan berpasangan antara elemen yang berdekatan (`str[j]` dan `str[j+1]`). Jika `str[j]` lebih besar dari `str[j+1]`, kedua karakter tersebut akan ditukar posisinya menggunakan `swap()`, dan sebuah flag `swapped` akan diatur ke `true`. Apabila setelah satu iterasi penuh dari *loop dalam* tidak ada pertukaran yang terjadi (artinya `swapped` tetap `false`), maka string dianggap sudah terurut sepenuhnya, dan *loop terluar* akan dihentikan lebih awal untuk meningkatkan efisiensi.

Kompleksitas waktu dari Bubble Sort tergantung pada kondisi awal data yang akan diurutkan. Pada kasus terbaik (best case) kompleksitas waktu Bubble Sort adalah $\mathcal{O}(n)$, karena data yang ada sudah terurut, algoritma yang ada hanya akan melakukan perbandingan antar elemen atau pengecekan. Pada kasus rata-rata (average case) dan kasus terburuk (worst case) kompleksitas waktunya adalah $\mathcal{O}(n^2)$, karena saat data terurut acak atau terbalik, algoritma yang ada harus membandingkan dan menggeser hampir semua

elemen. Sementara itu, kompleksitas ruang dari Insertion Sort adalah $O(1)$ karena algoritma Insertion Sort tidak memerlukan struktur data tambahan.

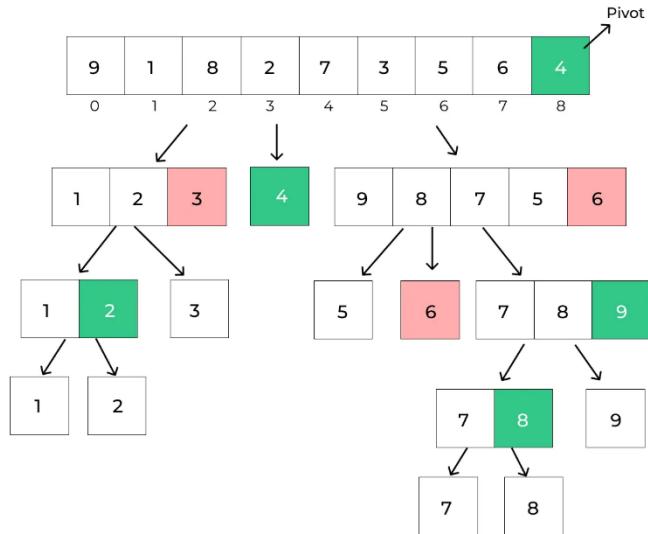
Jadi kesimpulannya, Insertion Sort ini sangat cocok digunakan untuk mengurutkan data jumlah yang kecil atau hampir terurut. Kemudian juga hemat ruang penyimpanan dalam penggunaannya karena tidak perlu tambahan struktur data dalam prosesnya, hanya beberapa variabel bantu sebagai penyimpanan sementara.

- **Quick Sort**

Quick Sort digunakan untuk melakukan sebuah pengurutan data, yang mana menggunakan pendekatan divide and conquer dalam proses. Cara kerja pengurutannya dimulai dengan memilih satu elemen sebagai pivot, biasanya elemen terakhir dalam kumpulan data. Selanjutnya, data dibagi menjadi dua bagian dimana elemen yang lebih kecil atau sama dengan pivot ditempatkan di sebelah kiri, sedangkan elemen yang lebih besar ditempatkan di sebelah kanan. Pivot kemudian diposisikan di tempat yang tepat dalam urutan akhir.

Proses pemisahan ini dilakukan oleh *fungsi partition()*, yang membandingkan setiap elemen dengan pivot. Bila ditemukan elemen yang lebih kecil atau sama dengan pivot, maka elemen itu akan ditukar ke posisi yang lebih kiri. Setelah semua elemen diperiksa, pivot akan ditukar ke posisi tengah yang sesuai, dan posisi ini akan menjadi batas pembagian data selanjutnya.

Setelah itu, *fungsi quickSort()* akan memanggil dirinya sendiri secara rekursif untuk mengurutkan bagian kiri dan kanan dari pivot. Jika masih ada bagian yang belum terurut, maka proses ini akan diulangi dengan memilih pivot baru dan membagi ulang datanya. Proses ini terus berlangsung hingga seluruh elemen berada di posisi yang tepat dan data menjadi terurut sempurna.



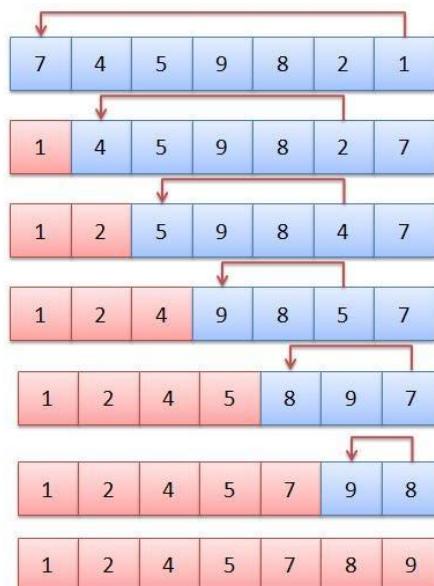
Gambar 114. Ilustrasi Quick Sort

Kompleksitas waktu dari Quick Sort tergantung pada pemilihan pivot dan kondisi awal data yang akan diurutkan. Pada kasus terbaik (best case) kompleksitas waktunya adalah $\Omega(n \log n)$, karena pembagian data menjadi dua bagian yang relatif seimbang. Pada kasus rata-rata (average case) kompleksitas waktu adalah $\Theta(n \log n)$, karena pembagian data biasanya cukup merata dalam banyak kasus. Pada kasus terburuk (worst case) kompleksitas waktunya menjadi $O(n^2)$, yaitu ketika pivot yang dipilih selalu menjadi elemen terkecil atau terbesar (misalnya saat data sudah terurut atau terbalik). Sementara itu, kompleksitas ruang dari Quick Sort pada kasus terbaik dan rata-rata adalah $O(\log n)$ karena membagi dua bagian dengan seimbang atau hampir seimbang. Sedangkan, untuk kompleksitas ruang dari Quick Sort pada kasus terburuk adalah $O(n)$ karena pembagiannya yang tidak seimbang.

Jadi kesimpulannya, Quick Sort ini sangat cocok digunakan untuk mengurutkan data jumlah yang besar karena hasil dari pengurutan datanya yang akurat dan waktu eksekusi rata-ratanya yang efisien. Kemudian, penggunaan ruang penyimpanannya tergolong kecil karena tidak memerlukan struktur data tambahan.

- **Selection Sort**

Selection Sort digunakan untuk melakukan sebuah pengurutan data, yang mana akan memilih elemen terkecil dari bagian data yang belum terurut, lalu menukarnya dengan elemen di posisi paling awal dari bagian tersebut. Proses ini dilakukan berulang-ulang hingga semua elemen yang ada berada diposisi yang sudah tepat atau terurut. Cara kerja pengurutannya dimulai dengan mencari elemen terkecil dari seluruh data, lalu menempatkannya di posisi pertama. Selanjutnya, algoritma melanjutkan pencarian elemen terkecil berikutnya dari data yang tersisa dan menukarnya ke posisi kedua, dan seterusnya. Dengan pendekatan ini, data secara bertahap akan terurut dari posisi awal hingga akhir melalui proses seleksi dan pertukaran.



Gambar 115. Ilustrasi Selection Sort

Dalam pembahasan fungsinya, `void selectionSort(string &str)` mengimplementasikan algoritma Selection Sort, yang bekerja dengan cara berulang kali mencari elemen terkecil dari bagian string yang belum terurut dan menempatkannya pada posisi yang benar. Ini dilakukan melalui `loop for` terluar yang mengiterasi dari awal string hingga satu posisi sebelum akhir. Dalam setiap iterasi, sebuah variabel `minIndex` diinisialisasi dengan indeks `i` saat ini, lalu `loop for` dalam akan mencari di sisa bagian string (`dari j = i + 1 hingga akhir`) untuk menemukan `indeks (j)` dari karakter yang paling kecil. Setelah `loop`

dalam selesai, elemen pada $str[i]$ dan elemen pada $str[minIndex]$ (yang merupakan elemen terkecil yang ditemukan) akan ditukar posisinya menggunakan $swap()$, sehingga elemen terkecil tersebut "dipilih" dan ditempatkan pada posisi yang benar di awal bagian yang belum terurut. Proses ini terus berlanjut hingga seluruh string terurut.

Kompleksitas waktu dari Selection Sort tidak bergantung pada kondisi awal data, baik data tersebut sudah terurut, hampir terurut, maupun acak sepenuhnya. Pada kasus terbaik (best case), kasus rata-rata (average case), dan kasus terburuk (worst case), kompleksitas waktunya tetap $O(n^2)$, karena algoritma harus mencari elemen terkecil dalam bagian yang belum terurut di setiap langkahnya, tanpa memperhatikan apakah data sudah dalam urutan tertentu. Sementara itu, kompleksitas ruang dari Selection Sort adalah $O(1)$, karena tidak perlu tambahan struktur data dalam prosesnya.

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 116. Soal 1 Sequential Searching Modul 6

- 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        cout << "Angka tidak ditemukan!";
65    return 0;
66 }
67

```

Gambar 117. Soal 1 Binary Searching Modul 6

- Tampilan Menu Program

```

Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih :

```

Gambar 118. Soal 1 Tampilan Menu Program Modul 6

Jelaskan perbedaan Sequential Searching dan Binary Searching beserta kelebihan dan kekurangan masing-masing?

A. Source Code

Tabel 9. Source Code Soal 1 Modul 6

```
1 #include <iostream>
2 #include <conio.h>
3 #include <random>
4 #include <vector>
5 #include <algorithm>
6
7 using namespace std;
8
9 void sequentialSearch(const vector<int> &nums,
10 int target)
11 {
12     vector<int> indices;
13
14     for (auto i = 0; i < nums.size(); i++)
15     {
16         if (nums[i] == target)
17             indices.push_back(i);
18
19         if (indices.empty())
20             cout << "angka " << target << " tidak ditemukan pada array" << endl;
21         else
22             cout << "Angka " << target << " terdapat pada array sebanyak " << indices.size() << " Kali" << endl;
23
24         cout << "Angka " << target << " ditemukan pada indeks: ";
25
26         for (auto i = 0; i < indices.size(); i++)
27         {
28             cout << indices[i];
29         }
30     }
31 }
```

```

25         if (i != indices.size() - 1) cout <<
26         ", ";
27         cout << "." << endl;
28     }
29 }
30
31 void binarySearch(const vector<int> &nums, int
32 target)
33 {
34     int high = nums.size() - 1;
35     int low = 0;
36     int index = -1;
37
38     while (low <= high)
39     {
40
41         int mid = low + (high - low) / 2;
42
43         if (nums[mid] == target)
44         {
45             index = mid;
46             break;
47         }
48         if (nums[mid] < target) low = mid + 1;
49         else high = mid - 1;
50     }
51
52     if (index == -1) cout << "angka " << target
53     << " tidak ditemukan pada array" << endl;
54     else cout << "angka " << target << " ditemukan
55     pada indeks ke " << index << endl;
56 }

```

```

53
54 void clearScreen()
55 {
56     system("cls");
57 }
58
59 void explain()
60 {
61     cout << "\nPERBEDAAN SEQUENTIAL/LINEAR SEARCH
DENGAN BINARY SEARCH" << endl;
62     cout << "SEQUENTIAL SEARCH: " << endl;
63     cout << "> Melakukan pengecekan pada array
melalui traversal indeks." << endl;
64     cout << "> Jika elemen pada array yang dicari
sama dengan elemen target, maka cetak nilai
indeks." << endl;
65     cout << "> Kompleksitas Waktu: O(n), karena
penggunaan fungsi loop for untuk pencarian target
secara traversal." << endl;
66     cout << "> Kompleksitas Ruang: O(1), karena
penggunaan memori konstan." << endl;
67     cout << "LINEAR SEARCH tidak memiliki syarat
tertentu." << endl;
68     cout << "LINEAR SEARCH dapat diterapkan pada:
" << endl;
69     cout << "1. Data yang Tidak Terurut" << endl;
70     cout << "2. Data berukuran kecil" << endl;
71     cout << "3. Pencarian Node Pada Linked List"
<< endl;
72     cout << endl;
73     cout << "BINARY SEARCH: " << endl;

```

```

74     cout << "> Membagi array menjadi dua bagian
melalui indeks tengah mid." << endl;
75     cout << "> Bandingkan elemen tengah dengan
elemen target." << endl;
76     cout << "> Jika elemen tengah sama dengan
elemen target, elemen pada array sudah ditemukan."
<< endl;
77     cout << "> Jika elemen tengah kurang dari
elemen target, cari di bagian kanan array." <<
endl;
78     cout << "> Jika elemen tengah lebih dari
elemen target, cari di bagian kiri array." <<
endl;
79     cout << "> Ulangi kedua tahap di atas sehingga
elemen target ditemukan." << endl;
80     cout << "> Kompleksitas Waktu: O(log n),
karena pembagian interval waktu pencarian." <<
endl;
81     cout << "> Kompleksitas Ruang: O(1), karena
penggunaan memori konstan." << endl;
82     cout << "SYARAT BINARY SEARCH: array harus
tersortir terlebih dahulu." << endl;
83     cout << "BINARY SEARCH dapat diterapkan pada:
" << endl;
84     cout << "1. Machine Learning" << endl;
85     cout << "2. Computer Graphics (algoritma untuk
ray tracing atau texture mapping)" << endl;
86     cout << "3. Pencarian data pada dataset besar"
<< endl;
87 }
88
89 int main()

```

```

90  {
91      int opt, target;
92      do
93      {
94          cout << "Pilih menu" << endl;
95          cout << "1. Sequential Searching" << endl;
96          cout << "2. Binary Searching" << endl;
97          cout << "3. Jelaskan Perbedaan Sequential
98          Searching dan Binary Searching!" << endl;
99          cout << "4. Exit" << endl;
100         cout << "Pilih: ";
101
102         cin >> opt;
103
104         switch (opt)
105         {
106             case 1:
107                 {
108                     vector<int> nums (100);
109                     mt19937_64
110                     rng(random_device{}());
111                     uniform_int_distribution<int>
112                     dist(1, 50);
113
114
115                     cout << "Generating 100
116                     numbers..." << endl;
117                     for (auto i = 0; i < nums.size();
118                         i++)

```

```

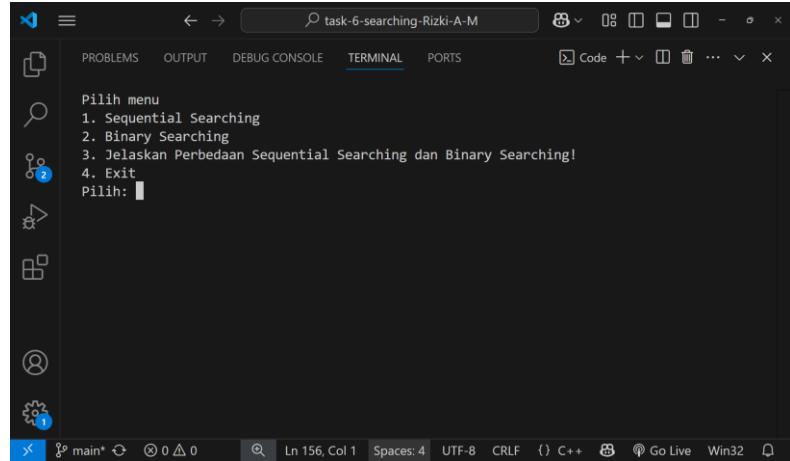
117         {
118             cout << nums[i] << "[" << i
119             << "]" << " ";
120         }
121         cout << endl;
122         cout << "Masukkan angka yang
123         ingin dicari: "; cin >> target;
124
125         sequentialSearch(nums, target);
126         break;
127     }
128
129     case 2:
130     {
131         int size;
132         cout << "Masukkan ukuran vector:
133         ";
134         cin >> size;
135         if (size < 1)
136         {
137             cout << "Error: Mohon
138             masukkan bilangan di atas 0" << endl;
139             break;
140         }
141         vector<int> nums(size);
142         mt19937_64
143         rng(random_device{}());
144         uniform_int_distribution<int>
145         dist(1, 100);
146
147         for (auto &val: nums)

```



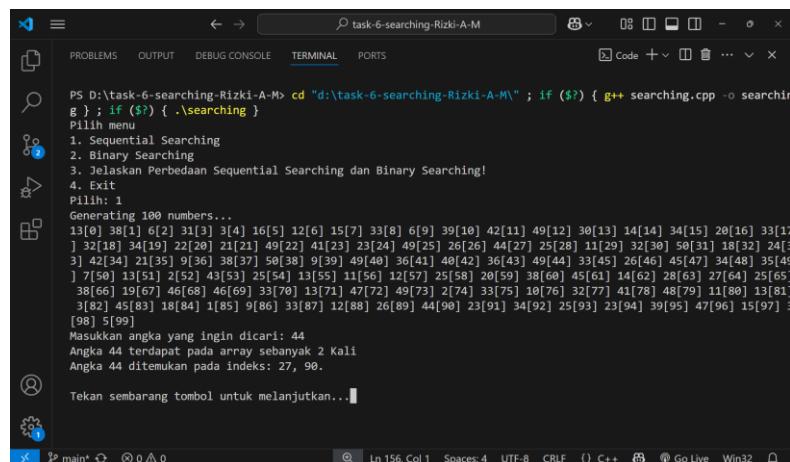
```
170         default:
171             cout << "Opsi tidak terdefinisi,
172             mohon masukkan ulang opsi" << endl;
173             break;
174         }
175         if (opt != 4)
176         {
177             cout << "\nTekan sembarang tombol
178             untuk melanjutkan...";
179             getch();
180             clearScreen();
181         }
182     }
183     while (opt != 4);
184
185     return 0;
186 }
```

B. Output Program



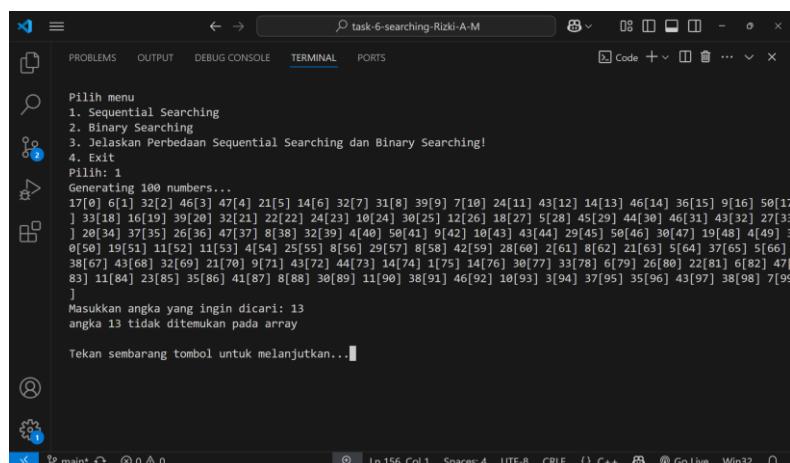
```
task-6-searching-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code + ... x
Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 1
```

Gambar 119. Tampilan Menu Searching



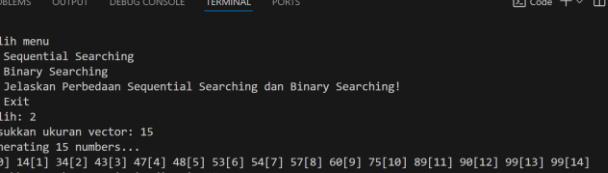
```
task-6-searching-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code + ... x
PS D:\task-6-searching-Rizki-A-M> cd "d:\task-6-searching-Rizki-A-M\"; if ($?) { g++ searching.cpp -o searching }
Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 1
Generating 100 numbers...
13[0] 38[1] 6[2] 31[3] 3[4] 16[5] 12[6] 15[7] 33[8] 6[9] 39[10] 42[11] 49[12] 30[13] 14[14] 34[15] 20[16] 33[17]
] 32[18] 34[19] 22[20] 21[21] 49[22] 41[23] 23[24] 49[25] 26[26] 44[27] 25[28] 11[29] 32[30] 50[31] 18[32] 24[33]
] 42[34] 21[35] 9[36] 38[37] 50[38] 9[39] 49[40] 36[41] 40[42] 36[43] 49[44] 33[45] 26[46] 45[47] 34[48] 35[49]
] 7[50] 13[51] 2[52] 43[53] 25[54] 13[55] 11[56] 12[57] 25[58] 20[59] 38[60] 45[61] 14[62] 28[63] 27[64] 25[65]
38[66] 19[67] 46[68] 46[69] 33[70] 13[71] 47[72] 49[73] 2[74] 33[75] 10[76] 32[77] 41[78] 48[79] 11[80] 13[81]
3[82] 49[83] 18[84] 1[85] 9[86] 33[87] 12[88] 26[89] 44[90] 23[91] 34[92] 25[93] 23[94] 39[95] 47[96] 15[97] 2
[98] 5[99]
Masukkan angka yang ingin dicari: 44
Angka 44 terdapat pada array sebanyak 2 kali
Angka 44 ditemukan pada indeks: 27, 90.
Tekan sembarang tombol untuk melanjutkan...
```

Gambar 120. Tampilan Sequential Searching Apabila Angka Ditemukan



```
task-6-searching-Rizki-A-M
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code + ... x
Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 1
Generating 100 numbers...
17[0] 6[1] 32[2] 46[3] 47[4] 21[5] 14[6] 32[7] 31[8] 39[9] 7[10] 24[11] 43[12] 14[13] 46[14] 36[15] 9[16] 50[17]
] 33[18] 16[19] 39[20] 32[21] 22[22] 24[23] 10[24] 30[25] 12[26] 18[27] 5[28] 45[29] 44[30] 46[31] 43[32] 27[33]
] 20[34] 37[35] 26[36] 47[37] 8[38] 32[39] 4[40] 50[41] 9[42] 10[43] 43[44] 29[45] 50[46] 30[47] 19[48] 4[49]
] 0[50] 19[51] 11[52] 11[53] 4[54] 25[55] 8[56] 29[57] 8[58] 42[59] 28[60] 2[61] 8[62] 21[63] 5[64] 37[65] 5[66]
38[67] 42[68] 32[69] 21[70] 9[71] 43[72] 44[73] 14[74] 1[75] 14[76] 30[77] 33[78] 6[79] 26[80] 22[81] 6[82] 47[82]
83] 11[84] 23[85] 35[86] 41[87] 8[88] 30[89] 11[90] 38[91] 46[92] 10[93] 3[94] 37[95] 35[96] 43[97] 38[98] 7[99]
]
Masukkan angka yang ingin dicari: 13
angka 13 tidak ditemukan pada array
Tekan sembarang tombol untuk melanjutkan...
```

Gambar 121. Tampilan Sequential Searching Apabila Angka Tidak Ditemukan



Pilih menu

1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit

Pilih: 2

Masukkan ukuran vector: 15

Generating 15 numbers...

7[0] 14[1] 34[2] 43[3] 47[4] 48[5] 53[6] 54[7] 57[8] 60[9] 69[10] 75[11] 89[12] 90[13] 99[14]

Masukkan angka yang ingin dicari: 47

angka 47 ditemukan pada indeks ke 4

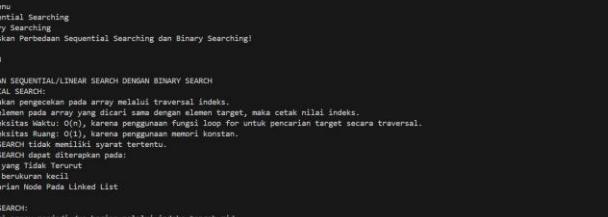
Tekan sembarang tombol untuk melanjutkan...■

Gambar 122. Tampilan Binary Searching Apabila Angka Ditemukan

```
Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 2
Masukkan ukuran vector: 10
Generating 10 numbers...
[2] 9[1] 19[2] 37[3] 42[4] 43[5] 43[6] 57[7] 68[8] 88[9]
Masukkan angka yang ingin dicari: 4
angka 4 tidak ditemukan pada array

Tekan sembarang tombol untuk melanjutkan...■
```

Gambar 123. Tampilan Binary Searching Apabila Angka Tidak Ditemukan



task 6-searching Rizki A M

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Plilih menu

- 1. Sequential Searching
- 2. Binary Searching
- 3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
- 4. Exit
- 5. Plilih 3

PERBEDAAN SEQUENTIAL/LINEAR SEARCH DENGAN BINARY SEARCH

SEQUENTIAL SEARCH:

- > Melakukan pencarian pada array melalui traversal indeks.
- > Jika ada elemen array yang dicari sama dengan elemen target, maka cetak nilai indeks.
- > Kompleksitas Waktu: $O(n)$, karena penggunaan Fungsi Loop for untuk pencarian target secara traversal.
- > Kompleksitas Ruang: $O(1)$, karena penggunaan memori konstan.

LINEAR SEARCH tidak memiliki syarat tertentu.

LINEAR SEARCH dapat diterapkan pada:

- 1. Data yang tidak terurut
- 2. Data berukuran kecil
- 3. Pencarian Node Pada Linked List

BINARY SEARCH:

- > Mengambil array melalui dua bagian melalui indeks tengah mid.
- > Bandingkan elemen tengah dengan elemen target.
- > Jika elemen tengah sama dengan elemen target, maka elemen pada array sudah ditemukan.
- > Jika elemen tengah lebih besar dari elemen target, cari di bagian kanan array.
- > Jika elemen tengah lebih dari elemen target, cari di bagian kiri array.
- > Ulangi keduah tahap di atas sehingga elemen target ditemukan.
- > Kompleksitas Waktu: $O(\log n)$, karena pembagian interval waktu pencarian.
- > Kompleksitas Ruang: $O(1)$, karena penggunaan memori konstan.

SYARAT BINARY SEARCH array harus terurut terlebih dahulu.

BINARY SEARCH dapat diterapkan pada:

- 1. Machine Learning
- 2. Computer Graphics (algoritma untuk ray tracing atau texture mapping)
- 3. Pencarian data pada dataset besar

Tekan sembarang tombol untuk melanjutkan... ↵

Gambar 124. Tampilan Penjelasan Perbedaan Dua Metode Search

C. Pembahasan

- **Alur Program**

Program Search apabila dimulai, akan muncul menu utama yang akan menampilkan 4 pilihan. Pilihan pertama untuk melakukan Sequential Search, pilihan kedua untuk melakukan Binary Search, pilihan ketiga menampilkan penjelasan perbedaan dari Sequential Search dan Binary Search, dan pilihan keempat untuk keluar dari program yang ada. Program yang ada akan terus muncul atau jalan karena adanya loop dan akan berhenti apabila pilihan keempat atau exit dipilih. Setelah setiap pilihan yang ada selesai melakukan apa yang diinginkan oleh user, tampilan yang ada akan otomatis dibersihkan dengan menekan tombol apapun yang ada agar tampilan terminal tetap rapi dan tidak menumpuk.

Apabila pilihan pertama yaitu Sequential Search yang dipilih, program yang ada akan secara otomatis membuat 100 angka acak dari nilai 1 sampai 50 dan menampilkannya dengan indeks. Kemudian akan muncul pilihan angka berapa yang akan di cari dari 100 angka acak yang ada. Setelah menentukan angkat yang ingin dicari, maka *fungsi sequentialSearch ()* akan bekerja dengan memindai setiap elemen dalam *vector* secara berurutan. Dan akhirnya hasil dari fungsi tersebut akan memberi tahu user kalo angka yang dicari ditemukan sebanyak sekian dan di indeks berapa saja angka tersebut ada, apabila angka yg ingin dicari tidak ada maka akan muncul pesan angka yang ingin dicari tidak ditemukan.

Apabila pilihan kedua yaitu Binary Search yang dipilih, program akan meminta user yang ada untuk menentukan ukuran dari *vektor* yang ingin dibuat. User harus memasukkan nilai lebih dari 0 agar program dapat lanjut ke tahap berikutnya, apabila user memasukkan besar *vektor* sama dengan 0 program akan menampilkan pesan Error dan kembali ke menu utama. Lanjut setelah selesai menentukan besar *vektor*, program akan secara otomatis membuat angka dari 1 sampai 100 sesuai dengan besar dari *vektor awal* dan ditampilkan dalam indeks. Kemudian user diminta untuk memasukkan angka berapa yang ingin dicari, setelahnya *fungsi binarySearch ()* akan dijalankan. *Fungsi binary search* bekerja

dengan membagi *vector* menjadi dua bagian secara berulang, sehingga lebih efisien untuk data yang sudah terurut. Dan akhirnya akan menampilkan angka yang dicari ada di indeks berapa dan apabila angka yang dicari tidak ada akan muncul pesan kalo angka yang dicari tidak ditemukan.

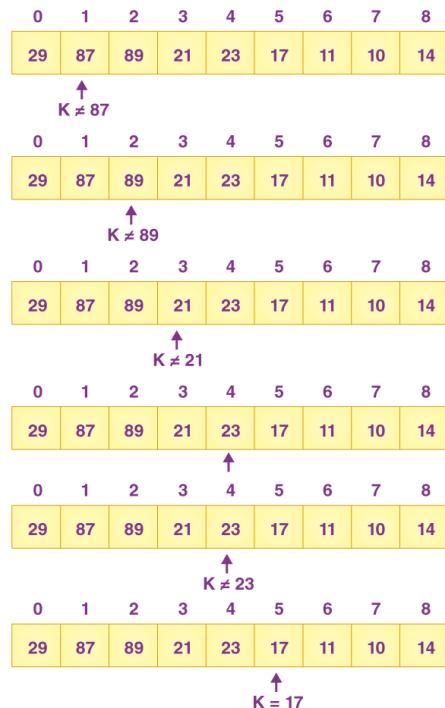
Apabila pilihan ketiga yang dipilih, akan muncul penjelasan dari perbedaan Sequential Search dan Binary Search. Penjelasan yang ada akan membahas cara kerja masing-masing, analisis kompleksitas waktu dan ruang serta kondisi yang cocok untuk penerapan algoritma yang ada.

Terus yang terakhir pilihan keempat yaitu exit, akan menampilkan pesan penutup untuk mengakhiri program yang ada.

- **Sequential Search**

Sequential Search, adalah metode yang bekerja dengan memeriksa setiap elemen dalam sebuah daftar secara berurutan hingga menemukan nilai yang dicari atau mencapai akhir daftar. Dalam implementasi ini, ketika elemen yang cocok dengan nilai target ditemukan, indeks (posisi) dari elemen tersebut akan disimpan ke dalam sebuah *vector* bernama *indices*.

Untuk menentukan apakah nilai target ada dalam daftar, program cukup memeriksa apakah *vector* *indices* kosong. Apabila *indices* kosong, ini menandakan bahwa nilai target tidak ditemukan sama sekali. Namun, apabila *indices* tidak kosong, program akan menampilkan berapa kali nilai target ditemukan (*menggunakan fungsi size()* dari *vector* *indices*), dan kemudian akan mencetak semua indeks tempat nilai target tersebut berada. Proses pencetakan indeks ini dilakukan dengan mengiterasi melalui setiap elemen di dalam *vector* *indices* menggunakan *loop for*.



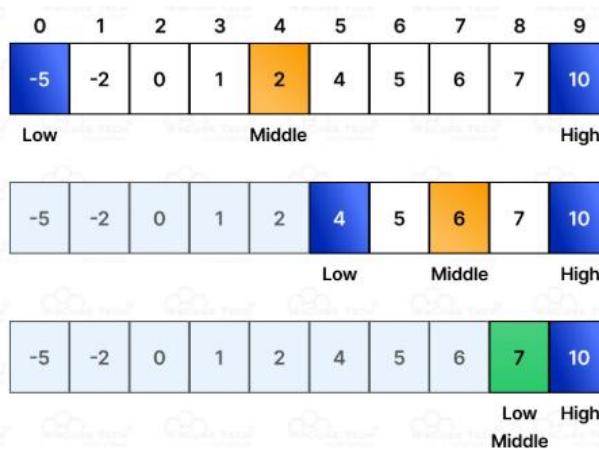
Gambar 125. Ilustrasi Sequential Search

Sequential Search memiliki kompleksitas waktu $O(n)$, dimana fungsi ini melakukan iterasi melalui setiap elemen dalam *vector* *nums* menggunakan *loop for*. Dalam skenario terburuk (misalnya, elemen target berada di akhir *vector* atau tidak ada sama sekali), fungsi ini harus memeriksa semua *n* elemen. Oleh karena itu, Semakin banyak jumlah elemen dalam vektor, maka waktu yang dibutuhkan juga semakin lama. Artinya, waktu prosesnya tumbuh sebanding dengan jumlah elemen, atau disebut juga $O(n)$ (linear). Sementara itu, Sequential Search memiliki kompleksitas ruang $O(n)$, dimana fungsi ini menggunakan *vector* *indices* untuk menyimpan semua indeks tempat angka target ditemukan. Dalam skenario terburuk, apabila semua elemen dalam *vector* *nums* adalah angka target yang dicari, maka *vector* *indices* akan menyimpan *n* indeks. Jadi, semakin banyak elemen dalam vektor input, maka semakin besar juga memori yang dibutuhkan. Ini berarti penggunaan memorinya sebanding dengan jumlah elemen, atau disebut $O(n)$ (linear).

- **Binary Search**

Binary Search adalah algoritma pencarian yang sangat efisien, dirancang khusus untuk menemukan elemen dalam sebuah daftar atau *array* yang sudah dalam kondisi terurut. Cara kerjanya adalah dengan terus-menerus mengurangi interval pencarian elemen target secara iteratif. Dalam kode ini, implementasi Binary Search menggunakan pendekatan iteratif melalui *loop while* untuk secara sistematis membagi ruang pencarian dan membandingkannya dengan elemen target.

Proses dimulai dengan mendefinisikan batas-batas pencarian, *high* diinisialisasi sebagai indeks paling kanan (*ukuran array* - 1), dan *low* sebagai indeks paling kiri (0). Sebuah variabel index juga disiapkan dengan *nilai* -1 sebagai penanda awal apabila target belum ditemukan. *Loop while* akan terus berjalan selama *low* tidak melebihi *high*. Di setiap iterasi, program menghitung *mid* (indeks tengah) dari interval pencarian saat ini. Nilai elemen di *mid* kemudian dibandingkan dengan elemen target. Apabila sama, pencarian berhasil dan index diperbarui. Apabila elemen di *mid* lebih kecil dari target, pencarian dilanjutkan di paruh kanan *array* dengan menggeser *low* ke *mid* + 1, dan apabila elemen di *mid* lebih besar dari target, pencarian dilanjutkan di paruh kiri *array* dengan menggeser *high* ke *mid* - 1. Proses pembagian dan perbandingan ini berulang hingga target ditemukan atau interval pencarian habis, menandakan target tidak ada di dalam *array*.



Gambar 126 Ilustrasi Binary Search

Binary Search memiliki kompleksitas waktu $O(\log n)$, dimana fungsi ini bekerja dengan membagi ruang pencarian menjadi dua pada setiap langkah iterasi (melalui variabel *mid*). Ini berarti jumlah langkah yang dibutuhkan untuk menemukan elemen target (atau menentukan bahwa elemen tidak ada) berkang secara eksponensial dengan setiap langkah. Misalnya, ada 1024 elemen, dibutuhkan maksimal sekitar 10 langkah ($\log_2 1024 = 10$). Jadi, semakin banyak elemen, waktu yang dibutuhkan memang bertambah, tapi tidak terlalu cepat atau hanya naik sedikit setiap kali jumlah elemen digandakan. Ini disebut $O(\log n)$ (logaritmik). Sementara itu, Binary Search memiliki kompleksitas ruang $O(1)$, dimana fungsi ini hanya menggunakan sejumlah kecil variabel tetap seperti *high*, *low*, *mid*, dan *index*, terlepas dari ukuran *vector* masukan. Ruang memori yang digunakan tidak bertambah seiring dengan peningkatan *n*, sehingga kompleksitas ruangnya adalah $O(1)$ (konstan).

- **Perbandingan Sequential Search Dengan Binary Search**

Pada dasarnya, perbedaan fundamental antara Sequential Search dan Binary Search terletak pada caranya untuk melakukan pencarian. Sequential Search bekerja dengan *metode traversal* yang sederhana, dimana ia memeriksa setiap elemen dalam *array* satu per satu, bergerak dari indeks paling kiri hingga paling kanan, hingga elemen target ditemukan. Di sisi lain, Binary Search menggunakan pendekatan yang jauh lebih efisien dengan membagi *array* menjadi dua bagian berdasarkan indeks tengah (*mid*). Setelah itu, ia membandingkan elemen tengah dengan target. Apabila sama, maka target ditemukan. Apabila elemen tengah kurang dari target, pencarian dilanjutkan di paruh kanan *array*, dan apabila elemen tengah lebih dari target, pencarian difokuskan pada paruh kiri *array*. Proses pembagian ini berulang hingga target ditemukan atau tidak ada lagi ruang pencarian.

Perbedaan cara kerja ini secara langsung memengaruhi kompleksitas waktu kedua algoritma. Sequential Search memiliki kompleksitas waktu $O(n)$, yang berarti waktu pencarian akan meningkat secara linear seiring dengan bertambahnya jumlah elemen (*n*) dalam *array*, karena ia

mungkin harus memeriksa setiap elemen. Hal ini disebabkan oleh iterasi *for loop* yang menyeluruh. Sebaliknya, Binary Search menunjukkan kompleksitas waktu yang jauh lebih efisien, yaitu $O(\log n)$. Efisiensi ini didapatkan karena pada setiap langkah, Binary Search secara efektif mengurangi ruang pencarian menjadi setengahnya, sehingga jumlah operasi yang diperlukan bertambah jauh lebih lambat dibandingkan ukuran *array*. Sementara itu, dalam hal kompleksitas ruang, baik Sequential Search maupun Binary Search umumnya dianggap memiliki kompleksitas $O(1)$ untuk variabel utama mereka, karena penggunaan memori tambahan bersifat konstan dan tidak bergantung pada ukuran *input* (meskipun sequential Search dalam kode memiliki *vector<int>* indices yang bisa berukuran $O(n)$ di worst case apabila semua elemen adalah target).

Perbedaan mendasar lainnya adalah syarat yang harus dipenuhi oleh kedua algoritma. Sequential Search tidak memiliki syarat khusus, artinya ia dapat berfungsi dengan baik pada data yang tidak terurut, menjadikannya pilihan yang fleksibel untuk berbagai kondisi. Namun, fleksibilitas ini datang dengan konsekuensi karena kompleksitas waktunya $O(n)$, Sequential Search tidak cocok untuk *dataset* berukuran besar. Penerapannya lebih sesuai untuk data yang tidak terurut, *array* berukuran kecil, atau pencarian *node spesifik* pada *linked list*.

Di sisi lain, Binary Search memiliki syarat ketat, ia hanya dapat bekerja pada *array* yang sudah tersortir. Jika data belum terurut, proses pengurutan awal (misalnya dengan Merge Sort atau Quick Sort yang memiliki kompleksitas waktu $O(n \log n)$) harus dilakukan terlebih dahulu, yang menambah biaya komputasi. Meskipun demikian, karena efisiensinya yang tinggi $O(\log n)$, Binary Search sangat relevan dan dapat diterapkan pada bidang-bidang seperti Machine Learning, Computer Graphics (untuk algoritma *ray tracing* atau *texture mapping*), serta pencarian data pada *dataset* besar yang sudah terurut.

MODUL 7 : TREE (POHON)

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    int main()
66    {
67        int pil, data;
68        Node *pohon;
69        pohon = NULL;
70        do
71        {
72            system("cls");
73            cout << "1. Tambah\n";
74            cout << "2. PreOrder\n";
75            cout << "3. inOrder\n";
76            cout << "4. PostOrder\n";
77            cout << "5. Exit\n";
78            cout << "\nPilihan : ";
79            cin >> pil;
80            switch (pil)
81            {
82                case 1:
83                    cout << "\n INPUT : ";
84                    cout << "\n -----";
85                    cout << "\n Data baru : ";
86                    cin >> data;
87                    tambah(&pohon, data);
88                    break;
89                case 2:
90                    cout << "PreOrder";
91                    cout << "\n-----\n";
92                    if (pohon != NULL)
93                    {
94                        preOrder(pohon);
95                    }
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 127. Soal 1 Modul 7

A. Source Code

Tabel 10. Source Code Soal 1 Modul 7

```
1 #include <iostream>
2 #include <conio.h>
3 #include <stdlib.h>
4
5 using namespace std;
6
7 struct Node
8 {
9     int data;
10    Node *left;
11    Node *right;
12 };
13
14 void insert(Node **root, int newData)
15 {
16     if (*root == nullptr)
17     {
18         Node *newNode;
19         newNode = new Node;
20
21         newNode -> data = newData;
22         newNode -> left = nullptr;
23         newNode -> right = nullptr;
24
25         *root = newNode;
26
27         cout << " Data has been added";
28     }
29
30     else if (newData < (*root) -> data)
31     {
```

```

32         insert(&((*root)->left), newData);
33     }
34
35     else if (newData > (*root) -> data)
36     {
37         insert(&((*root)->right), newData);
38     }
39
40     else if (newData == (*root) -> data)
41     {
42         cout << " Data is already exist";
43     }
44 }
45
46 void preOrder(Node *root)
47 {
48     if (root == nullptr) return;
49     cout << root->data << " ";
50     preOrder(root->left);
51     preOrder(root->right);
52 }
53
54 void inOrder(Node *root)
55 {
56     if (root == nullptr) return;
57     inOrder(root->left);
58     cout << root->data << " ";
59     inOrder(root->right);
60 }
61
62 void postOrder(Node *root)
63 {

```

```

64     if (root == nullptr) return;
65     postOrder(root->left);
66     postOrder(root->right);
67     cout << root->data << " ";
68 }
69
70 // side quest
71 void printTree() {
72 }
73
74 void freeTree(Node *root)
75 {
76     if (root == nullptr) return;
77     freeTree(root->left);
78     freeTree(root->right);
79     delete root;
80 }
81
82 int main()
83 {
84     int opt, val;
85     Node *tree;
86     tree = nullptr;
87
88     do
89     {
90         system("cls");
91         cout << "1. Insert\n";
92         cout << "2. PreOrder\n";
93         cout << "3. InOrder\n";
94         cout << "4. PostOrder\n";
95         cout << "5. Exit\n";

```

```

96         cout << "\nOption: "; cin >> opt;
97         switch (opt)
98         {
99
100            case 1:
101                cout << "\n Input:";
102                cout << "\n -----";
103                cout << "\n New data: ";
104                cin >> val;
105                insert(&tree, val);
106                break;
107
108            case 2:
109                cout << "PreOrder
110 Traversal\n";
111                cout << "=====\\n";
112                if (tree == nullptr)
113                {
114                    cout << "Tree is
115 empty!\\n";
116                }
117                else
118                {
119                    preOrder(tree);
120                }
121                break;
122
123            case 3:
124                cout << "InOrder
125 Traversal\\n";

```

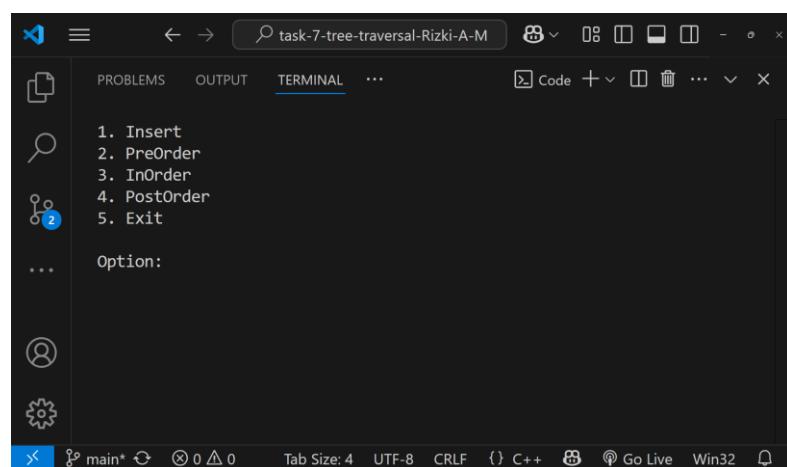
```

124           cout <<
125           "=====\\n";
126           if (tree == nullptr)
127           {
128               cout << "Tree is
129               empty!\\n";
130           }
131           else
132           {
133               inOrder(tree);
134           }
135           break;
136       case 4:
137           cout << "PostOrder
138           Traversal\\n";
139           cout <<
140           "=====\\n";
141           if (tree == nullptr)
142           {
143               cout << "Tree is
144               empty!\\n";
145           }
146           else
147           {
148               postOrder(tree);
149           }
150           break;
151       case 5:
152           freeTree(tree);

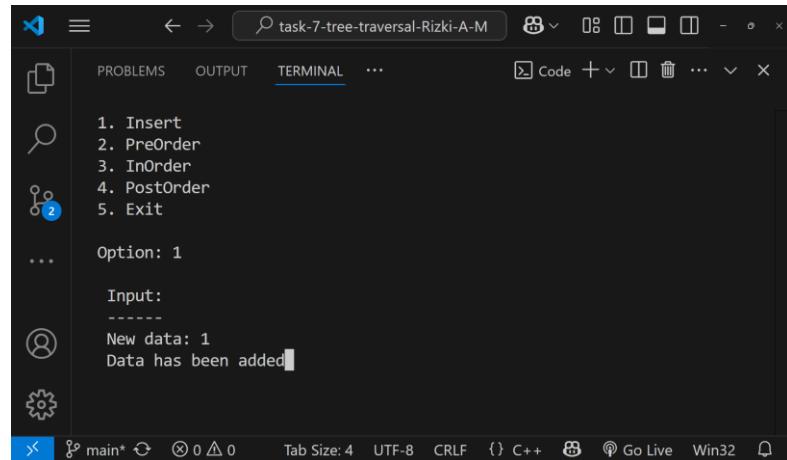
```

```
151             return 0;  
152  
153         default:  
154             cout << "Option is not valid!  
155             Please re-enter your option";  
156             break;  
157         }  
158     }  
159     while(opt != 5);  
160     return 0;  
161 }
```

B. Output Program

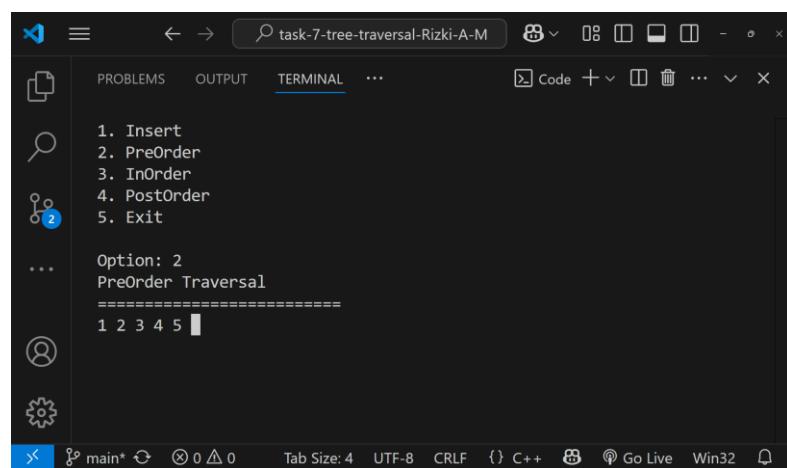


Gambar 128. Tampilan Menu Tree



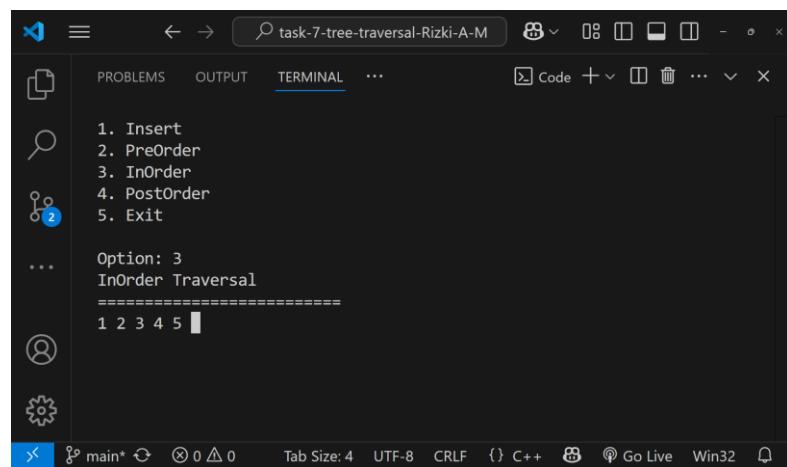
```
task-7-tree-traversal-Rizki-A-M
PROBLEMS OUTPUT TERMINAL ...
1. Insert
2. PreOrder
3. InOrder
4. PostOrder
5. Exit
...
Option: 1
Input:
-----
New data: 1
Data has been added
```

Gambar 129. Tampilan Menu Insert Tree



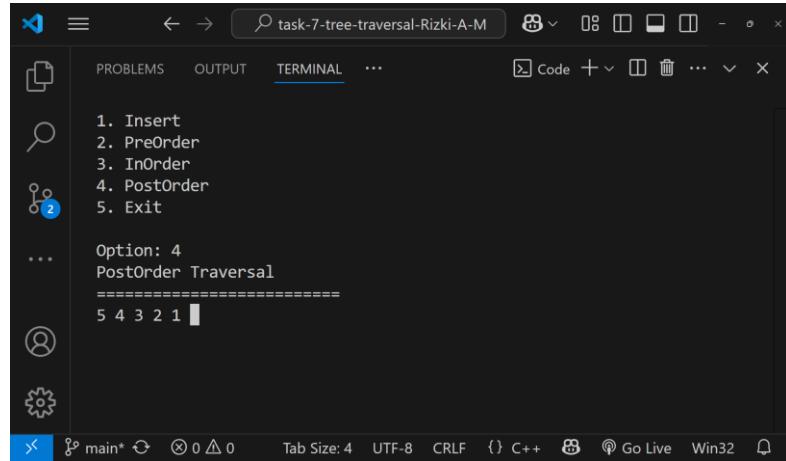
```
task-7-tree-traversal-Rizki-A-M
PROBLEMS OUTPUT TERMINAL ...
1. Insert
2. PreOrder
3. InOrder
4. PostOrder
5. Exit
...
Option: 2
PreOrder Traversal
=====
1 2 3 4 5
```

Gambar 130. Tampilan Menu PreOrder Tree



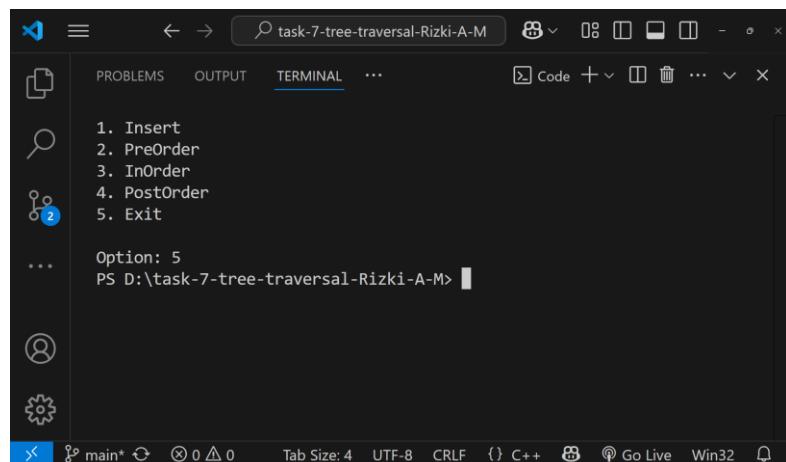
```
task-7-tree-traversal-Rizki-A-M
PROBLEMS OUTPUT TERMINAL ...
1. Insert
2. PreOrder
3. InOrder
4. PostOrder
5. Exit
...
Option: 3
InOrder Traversal
=====
1 2 3 4 5
```

Gambar 131. Tampilan Menu InOrder Tree



```
task-7-tree-traversal-Rizki-A-M
PROBLEMS OUTPUT TERMINAL ...
1. Insert
2. PreOrder
3. InOrder
4. PostOrder
5. Exit
...
Option: 4
PostOrder Traversal
=====
5 4 3 2 1
```

Gambar 132. Tampilan Menu PostOrder Tree



```
task-7-tree-traversal-Rizki-A-M
PROBLEMS OUTPUT TERMINAL ...
1. Insert
2. PreOrder
3. InOrder
4. PostOrder
5. Exit
...
Option: 5
PS D:\task-7-tree-traversal-Rizki-A-M>
```

Gambar 133. Tampilan Menu Exit

C. Pembahasan

• Alur Program

Program ini dimulai dengan mendefinisikan *struktur Node*, yang menjadi dasar pembentuk pohon biner. Setiap *Node* terdiri dari sebuah *variabel integer* yang akan menyimpan nilai, serta dua *pointer left* dan *right* yang masing-masing menunjuk ke anak kiri dan anak kanan dari *node* tersebut. Pada awal program, *pointer tree* akan diinisialisasi sebagai *nullptr* yang artinya pohon biner dalam keadaan kosong.

Selanjutnya, program akan menampilkan *menu utama* kepada pengguna dalam sebuah *loop do while*. Menu yang ada akan menyediakan lima opsi, yaitu *Insert* (untuk menambahkan data baru), *PreOrder* (untuk menampilkan data dengan urutan *pre-order traversal*),

InOrder (untuk menampilkan data dengan urutan *in-order traversal*), *PostOrder* (untuk menampilkan data dengan urutan *post-order traversal*), dan *Exit* (untuk keluar dari program). Setiap selesai menjalankan fungsi yang ada di setiap pilihan maka tampilan layar program akan dibersihkan dengan *system ("cls")* sehingga tidak adanya penumpukan hasil output pada layar. *Loop* akan terus berulang selama nilai variabel pilihan tidak sama dengan 5, dan setelah selesai menjalankan program yang ada (kecuali keluar), program akan menunggu input tombol apa pun karena adanya *getch()* sebelum akhirnya akan dibersihkan oleh *system ("cls")*.

Pemilihan opsi menu diatur menggunakan *switch case*. Apabila pengguna memilih *opsi 1* yaitu (*Insert*), program akan meminta input data baru yang kemudian disimpan dalam variabel *val*. Data ini lalu disisipkan ke dalam pohon melalui pemanggilan *fungsi insert()*. *Fungsi insert()* bekerja secara rekursif, yang mana apabila *root* saat ini adalah *nullptr*, *node* baru akan dibuat dan menjadi *root* di posisi tersebut. Jika data baru lebih kecil dari data pada *node root* saat ini, penyisipan akan dilanjutkan secara rekursif ke sub-pohon kiri. Sebaliknya, jika data baru lebih besar, penyisipan akan dilanjutkan ke sub-pohon kanan. Apabila data yang dimasukkan sudah ada, program akan menampilkan pesan "Data is already exist".

Untuk *opsi 2 (PreOrder)*, *opsi 3 (InOrder)*, dan *opsi 4 (PostOrder)*, program terlebih dahulu memeriksa apakah tree kosong (*nullptr*). Apabila pohon yang ada kosong, pesan "Tree is empty!" akan ditampilkan. Kemudian, apabila pohon yang ada berisi data, *fungsi traversal* yang sesuai akan dipanggil untuk *preOrder()*, *inOrder()*, atau *postOrder()*. *Fungsi preOrder()* akan menampilkan data node saat ini, lalu mengunjungi anak kiri, dan kemudian anak kanan. *Fungsi inOrder()* akan mengunjungi anak kiri, lalu menampilkan data node saat ini, dan kemudian mengunjungi anak kanan, menghasilkan urutan data yang terurut. Sementara itu, *fungsi*

postOrder() akan mengunjungi anak kiri, lalu anak kanan, dan terakhir menampilkan data node saat ini.

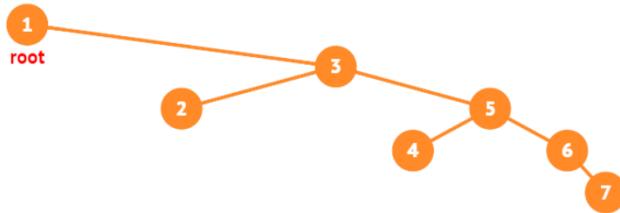
Terakhir, saat pengguna memilih *opsi 5 (Exit)*, program akan memanggil fungsi *freeTree()*. Fungsi ini bertanggung jawab atas dealokasi memori yang digunakan oleh pohon biner. *freeTree()* bekerja secara rekursif dengan pendekatan *post-order traversal*, yaitu membebaskan sub-pohon kiri terlebih dahulu, lalu sub-pohon kanan, dan barulah menghapus (*delete*) node saat ini. Proses ini memastikan bahwa semua memori yang dialokasikan secara dinamis untuk node-node pohon dikembalikan ke sistem, mencegah memory leak. Setelah seluruh memori dibebaskan, program akan keluar dengan mengembalikan nilai 0. Apabila pengguna memilih opsi yang tidak valid, program akan menampilkan pesan kesalahan "Option is not valid! Please re-enter your option".

- **Insertion**

Fungsi penting untuk program ini adalah *insert()*, yang bertanggung jawab untuk menambahkan data baru ke dalam pohon. *insert()* akan diinisialisasikan dengan Binary Search Tree (BST) yaitu sebuah struktur data yang efisien untuk pencarian dan pengurutan. Fungsi ini akan mengambil dua parameter yaitu sebuah *pointer ke pointer Node* yang merepresentasikan akar (atau sub-akar) pohon saat ini, dan sebuah *integer newData* yang merupakan data yang ingin disisipkan.

Proses penyisipan dimulai dengan memeriksa apakah posisi *root* saat ini *kosong (null)*. Apabila iya, berarti kita telah menemukan tempat yang tepat untuk *newData*. Maka, sebuah *newNode* akan dibuat, *newData* disematkan di dalamnya, dan pointer *left* serta *right* dari *newNode* diatur ke *nullptr* (karena ini adalah node daun baru). *newNode* ini kemudian menjadi *root* di posisi tersebut. Kemudian, apabila *root* yang ada tidak kosong, fungsi akan membandingkan *newData* dengan data yang ada di *root* saat ini. Jika *newData* lebih kecil, program akan memanggil *insert()* secara rekursif untuk *subtree kiri*. Sebaliknya, jika *newData* lebih besar, rekursi akan

terjadi pada *subtree kanan*. Ada hal penting disini yaitu bila *newData* sama dengan data yang sudah ada, program akan mengeluarkan pesan bahwa data duplikat, ini berguna untuk menjaga integritas BST sesuai aturan standar.

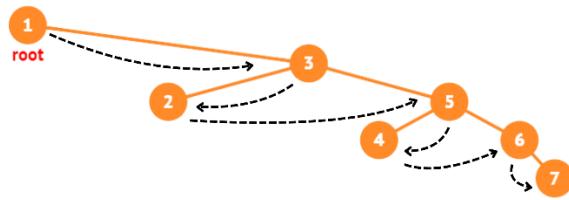


Gambar 134. Ilustrasi Insertion Pada Tree

- **Pre-Order Traversal**

Fungsi `preOrder()` berfungsi untuk melakukan traversal pada struktur data pohon (*tree*). Dalam metode ini, urutan kunjungan dimulai dari node induk (*parent*) terlebih dahulu, kemudian lanjut ke *subtree kiri*, dan yang terakhir ke *subtree kanan*. Traversal dilakukan secara rekursif, yang berarti fungsi akan terus memanggil dirinya sendiri selama masih ada node yang perlu dikunjungi. Langkah pertama dalam fungsi ini adalah memeriksa apakah node saat ini bernilai null. Apabila iya, maka tidak ada lagi node yang bisa dikunjungi dan fungsi akan berhenti.

Namun, apabila node yang ada tidak null, maka data atau nilai dari node tersebut langsung dicetak. Setelah itu, fungsi akan masuk ke subtree sebelah kiri dan melakukan proses yang sama secara rekursif. Setelah seluruh node di subtree kiri selesai dikunjungi, fungsi akan berpindah ke subtree kanan dan kembali menjalankan proses yang sama. Dengan pola tersebut, `preOrder()` selalu mengunjungi node dalam urutan: node saat ini, subtree kiri, dan subtree kanan. Pola ini berguna, misalnya, ketika ingin menyalin struktur pohon atau mengekstrak informasi dari atas ke bawah secara teratur.

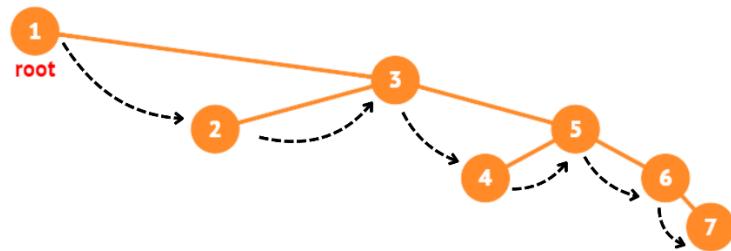


Gambar 135 Ilustrasi PreOrder Pada Tree

- **In-Order Traversal**

Fungsi `inOrder()` digunakan untuk melakukan penelusuran (*traversal*) pada struktur pohon. Pada metode ini, proses kunjungan dimulai dari *subtree kiri* terlebih dahulu, kemudian ke node induk (*parent*), dan terakhir ke *subtree kanan* secara rekursif. Urutan ini sangat berguna karena pada tree bertipe binary search tree (BST), traversal in-order akan mencetak data dalam urutan yang terurut dari yang terkecil hingga terbesar.

Saat fungsi dijalankan, hal pertama yang dilakukan adalah memeriksa apakah node saat ini bernilai null. Apabila `null`, berarti tidak ada *node* lagi yang bisa dikunjungi, sehingga fungsi akan berhenti. Namun, apabila terdapat *node*, maka fungsi akan lebih dulu mengunjungi *subtree kiri* secara rekursif. Setelah itu, data dari *node* induk akan dicetak. Terakhir, fungsi akan mengunjungi *subtree kanan* dan mencetak datanya. Pola kunjungannya mengikuti urutan dari kiri ke induk dan ke kanan, dan ini berulang hingga seluruh node telah dikunjungi.

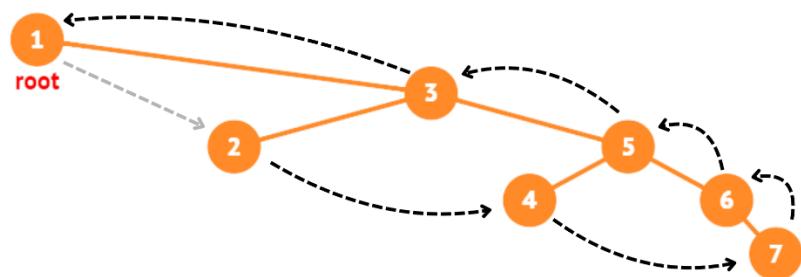


Gambar 136. Ilustrasi InOrde Pada Tree

- **Post-Order Traversal**

Fungsi `postOrder()` digunakan untuk melakukan traversal pada struktur pohon. Dalam metode ini, urutan kunjungannya dimulai dari *subtree kiri*, lalu ke *subtree kanan*, dan terakhir ke node induk (*parent*) secara rekursif, dan cocok digunakan ketika kita ingin memproses atau menghapus node setelah semua turunannya selesai dikunjungi, seperti saat menghapus seluruh isi pohon.

Cara kerjanya dimulai dengan memeriksa apakah node saat ini bernilai `null`. Apabila `null`, maka tidak ada *node* lagi yang bisa dikunjungi, maka fungsi akan berhenti. Namun apabila *node* ada, maka program pertama-tama akan mengunjungi *subtree kiri* secara rekursif. Setelah itu, program lanjut ke *subtree kanan* dan melakukan hal yang sama. Jika kedua sisi sudah selesai dikunjungi, barulah program mencetak data dari *node induk*. Dengan demikian, urutan traversal-nya adalah dari kiri ke kanan dan akhirnya ke induk, dan pola ini berulang hingga seluruh node dalam tree selesai dikunjungi.



Gambar 137. Ilustrasi PostOrder Pada Tree

TAUTAN GIF

Berikut adalah tautan untuk semua source code yang telah dibuat:

https://github.com/Rizki-A-M/Rizki-A-M-PRAKTIKUM_ALGORITMA_DAN_STRUKTUR_DATA.git