

Aturan Praktikum Struktur Data

1. **Akun GitHub:** Setiap praktikan wajib memiliki akun GitHub yang aktif dan digunakan selama praktikum berlangsung.
2. **Invite Collaborator:** Setiap praktikan diwajibkan untuk menambahkan collaborator di setiap repository
 - a. Asisten Praktikum: AndiniNH
 - b. Asisten Praktikum: 4ldiputra
3. **Repository Praktikum:** Setiap praktikan diwajibkan untuk membuat satu repository di GitHub yang akan digunakan untuk seluruh tugas dan laporan praktikum. Repository ini harus diatur dengan rapi dan sesuai dengan instruksi yang akan diberikan di lampiran.
4. **Penamaan Folder:** Penamaan folder dalam repository akan dibahas secara rinci di lampiran. Praktikan wajib mengikuti aturan penamaan yang telah ditentukan.

Nomor	Pertemuan	Penamaan
1	Pengalan Bahasa C++ Bagian Pertama	01_Pengenalan_CPP_Bagian_1
2	Pengenalan Bahasa C++ Bagian Kedua	02_Pengenalan_CPP_Bagian_2
3	Abstract Data Type	03_Abstract_Data_Type
4	Single Linked List Bagian Pertama	04_Single_Linked_List_Bagian_1
5	Single Linked List Bagian Kedua	05_Single_Linked_List_Bagian_2
6	Double Linked List Bagian Pertama	06_Double_Linked_List_Bagian_1
7	Stack	07_Stack
8	Queue	08_Queue
9	Assessment Bagian Pertama	09_Assessment_Bagian_1
10	Tree Bagian Pertama	10_Tree_Bagian_1
11	Tree Bagian Kedua	11_Tree_Bagian_2
12	Asistensi Tugas Besar	12_Asistensi_Tugas_Besar
13	Multi Linked List	13_Multi_Linked_List
14	Graph	14_Graph
15	Assessment Bagian Kedua	15_Assessment_Bagian_2
16	Tugas Besar	16_Tugas_Besar

5. Jam Praktikum:

- Jam masuk praktikum adalah **1 jam lebih lambat** dari jadwal yang tercantum. Sebagai contoh, jika jadwal praktikum adalah pukul 06.30 - 09.30, maka aturan praktikum akan diatur sebagai berikut:
 - **06.30 - 07.30:** Waktu ini digunakan untuk **Tugas Praktikum dan Laporan Praktikum** yang dilakukan di luar laboratorium.
 - **07.30 - 09.30:** Sesi ini mencakup **tutorial, diskusi, dan kasus problem-solving**. Kegiatan ini berlangsung di dalam laboratorium dengan alokasi waktu sebagai berikut:
 - **60 menit pertama:** Tugas terbimbing.
 - **60 menit kedua:** Tugas mandiri.

6. **Pengumpulan Tugasn Pendahuluan:** Tugas Pendahuluan (TP) wajib dikumpulkan melalui GitHub sesuai dengan format berikut:

nama_repo/nama_pertemuan/TP_Pertemuan_Ke.md

Sebagai contoh:

STD_Yudha_Islalmi_Sulistya_XXXXXXXX/01_Running_Modul/TP_01.md

7. **Pengecekan Tugas Pendahuluan:** Pengumpulan laporan praktikum akan diperiksa **1 hari sebelum praktikum selanjutnya** dimulai. Pastikan tugas telah diunggah tepat waktu untuk menghindari sanksi.

8. **Struktur Laporan Praktikum**

1. **Cover :**

LAPORAN PRAKTIKUM
Modul 5
“SINGLE LINKED LIST (BAGIAN KEDUA) ”



Disusun Oleh:
Reza Afiansyah Wibowo -2311104062
SE0702

Dosen :
Arief Rais bahtiar

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

Tujuan

Memahami penggunaan linked list dengan pointer operator- operator dalam program.

Memahami operasi-operasi dasar dalam linked list.

Membuat program dengan menggunakan linked list dengan prototype yang ada

Landasan Teori

Searching merupakan operasi dasar list dengan melakukan aktivitas pencarian terhadap node

tertentu. Proses ini berjalan dengan mengunjungi setiap node dan berhenti setelah node yang dicari

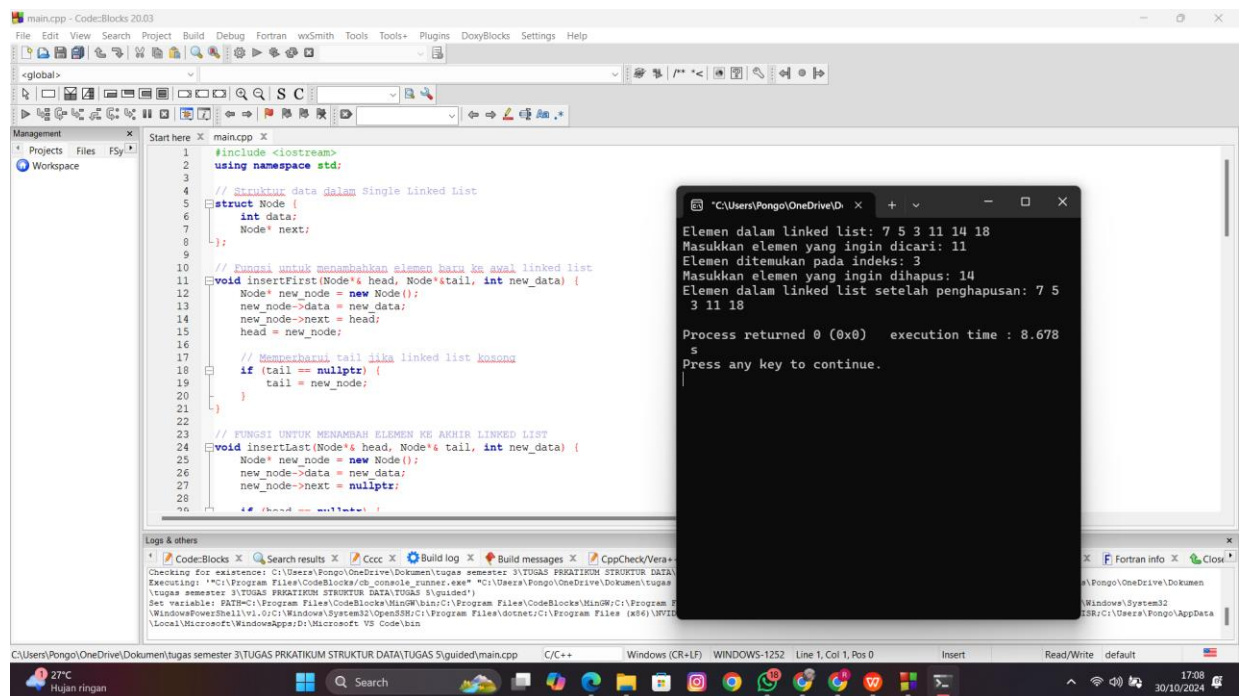
ketemu. Dengan melakukan operasi searching, operasi-operasi seperti insert after, delete after, dan

update akan lebih mudah.

Semua fungsi dasar diatas merupakan bagian dari ADT dari single linked list, dan aplikasi pada bahasa

pemrograman Cp semua ADT tersebut tersimpan dalam file *.c dan file *.h.

Guided



The screenshot shows the Code::Blocks IDE with a C++ project. The main.cpp file contains the following code:

```

1 #include <iostream>
2 using namespace std;
3
4 // Struktur data dalam Single Linked List
5 struct Node {
6     int data;
7     Node* next;
8 };
9
10 // Fungsi untuk menambahkan elemen baru ke awal linked list
11 void insertFirst(Node*& head, Node*&tail, int new_data) {
12     Node* new_node = new Node();
13     new_node->data = new_data;
14     new_node->next = head;
15     head = new_node;
16 }
17
18 // Menghapus tail jika linked list kosong
19 if (tail == nullptr) {
20     tail = new_node;
21 }
22
23 // Fungsi untuk menambah elemen ke akhir linked list
24 void insertLast(Node*& head, Node*&tail, int new_data) {
25     Node* new_node = new Node();
26     new_node->data = new_data;
27     new_node->next = nullptr;
28     if (head == nullptr) {
29         head = new_node;
30     }
31 }

```

The execution output shows the following steps:

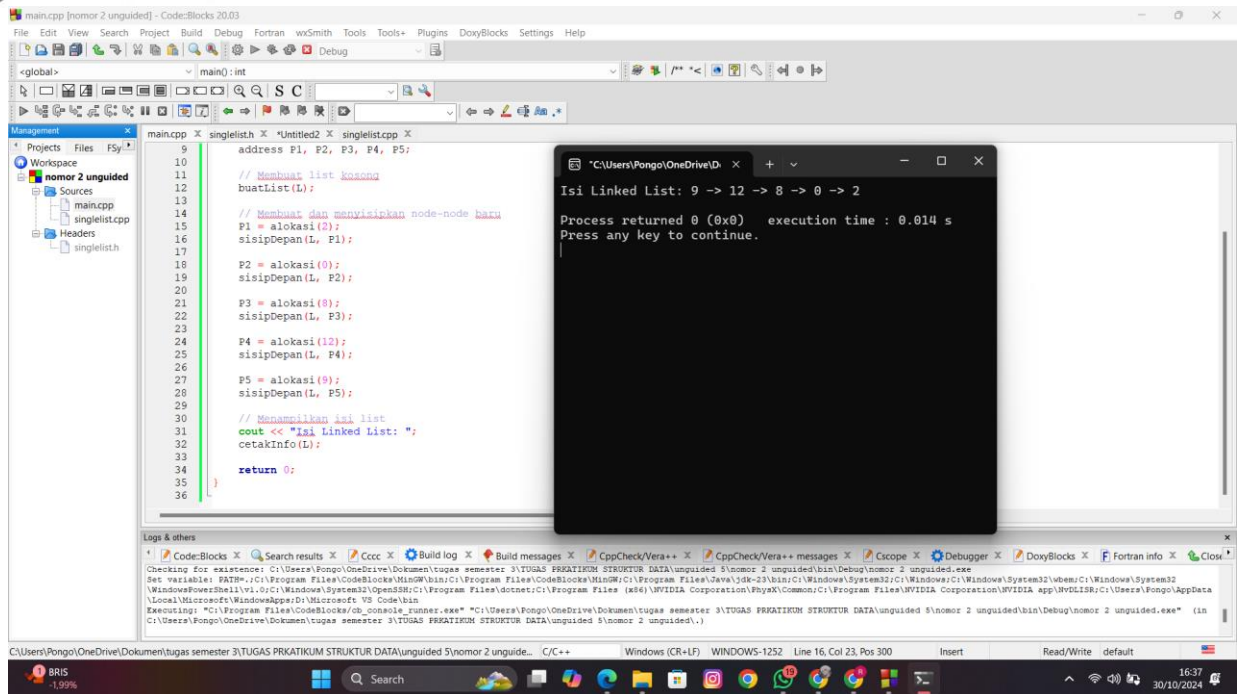
```

Elemen dalam linked list: 7 5 3 11 14 18
Masukkan elemen yang ingin dicari: 11
Elemen ditemukan pada indeks: 3
Masukkan elemen yang ingin dihapus: 14
Elemen dalam linked list setelah penghapusan: 7 5
3 11 18
Process returned 0 (0x0)   execution time : 8.678
s
Press any key to continue.

```

Unguided

Nomor2

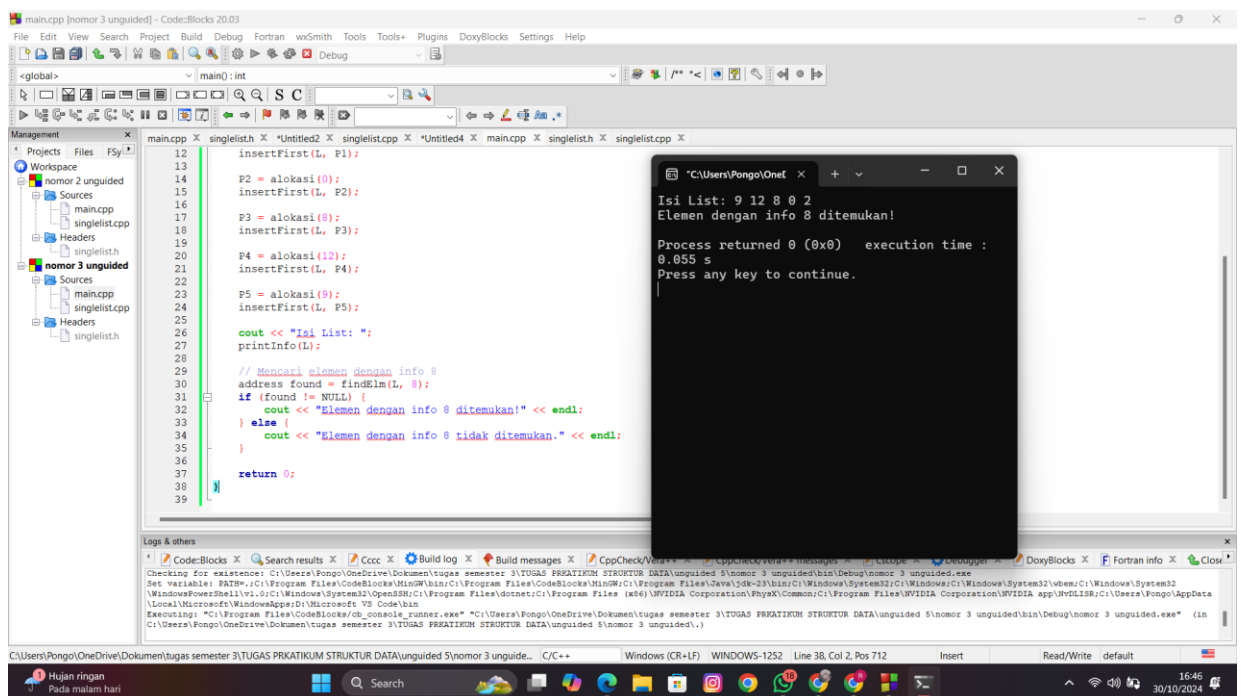


```

9      address P1, P2, P3, P4, P5;
10
11      // Membuat list kosong
12      buatList(L);
13
14      // Membuat dan menyisipkan node-node baru
15      P1 = alokasi(2);
16      sisipDepan(L, P1);
17
18      P2 = alokasi(9);
19      sisipDepan(L, P2);
20
21      P3 = alokasi(8);
22      sisipDepan(L, P3);
23
24      P4 = alokasi(12);
25      sisipDepan(L, P4);
26
27      P5 = alokasi(0);
28      sisipDepan(L, P5);
29
30      // Menampilkan isi list
31      cout << "Isi Linked List: ";
32      cetakInfo(L);
33
34      return 0;
35
36  }
    
```

Output: Isi Linked List: 9 -> 12 -> 8 -> 0 -> 2

Nomor 3

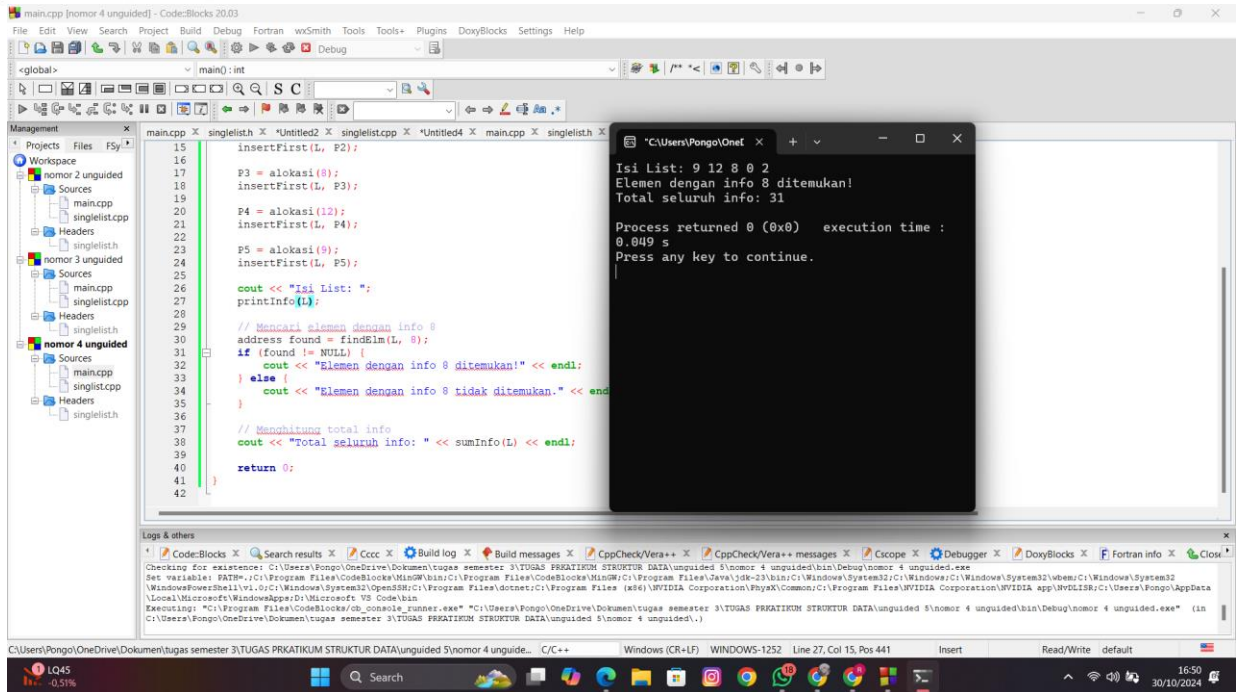


```

12      insertFirst(L, P1);
13
14      P2 = alokasi(9);
15      insertFirst(L, P2);
16
17      P3 = alokasi(8);
18      insertFirst(L, P3);
19
20      P4 = alokasi(12);
21      insertFirst(L, P4);
22
23      P5 = alokasi(0);
24      insertFirst(L, P5);
25
26      cout << "Isi List: ";
27      printInfo(L);
28
29      // Mencari elemen dengan info 8
30      address found = findElm(L, 8);
31      if (found != NULL) {
32          cout << "Elemen dengan info 8 ditemukan!" << endl;
33      } else {
34          cout << "Elemen dengan info 8 tidak ditemukan." << endl;
35      }
36
37      return 0;
38
39  }
    
```

Output: Isi List: 9 12 8 0 2
Elemen dengan info 8 ditemukan!

Nomor 4



The screenshot shows the Code::Blocks IDE with a C++ project named 'main.cpp [nomor 4 unguided]'. The code implements a linked list with functions for inserting, finding, and summing elements. The execution output in the console window shows the following results:

```
Isi List: 9 12 8 0 2
Elemen dengan info 8 ditemukan!
Total seluruh info: 31

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

The code in the editor is as follows:

```
15 insertFirst(L, P2);
16
17 P3 = alokasi(8);
18 insertFirst(L, P3);
19
20 P4 = alokasi(12);
21 insertFirst(L, P4);
22
23 P5 = alokasi(9);
24 insertFirst(L, P5);
25
26 cout << "Isi List: ";
27 printInfo(L);
28
29 // Mencari elemen dengan info 8
30 address found = findElm(L, 8);
31 if (found != NULL) {
32     cout << "Elemen dengan info 8 ditemukan!" << endl;
33 } else {
34     cout << "Elemen dengan info 8 tidak ditemukan." << endl;
35 }
36
37 // Menghitung total info
38 cout << "Total seluruh info: " << sumInfo(L) << endl;
39
40 return 0;
41
42 }
```

Kesimpulan

dokumen ini berfungsi sebagai panduan bagi mahasiswa atau pembelajar untuk memahami dan mengimplementasikan struktur data linked list dengan operasi dasar serta latihan dalam bahasa C.