

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	Pengalaman 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:
KAFKA PUTRA RIYADI - 2311104041
Kelas:
SE 07-02

Dosen :
Wahyu Andi Saputra, S.Pd., M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

2. Tujuan

1. Memahami penggunaan *linked list* dengan *pointer* operator- operator dalam program.
2. Memahami operasi-operasi dasar dalam *linked list*.
3. Membuat program dengan menggunakan *linked list* dengan *prototype* yang ada

3. Landasan Teori

Searching adalah operasi dasar dalam ilmu komputer yang digunakan untuk menemukan elemen tertentu atau sekelompok elemen dalam suatu struktur data. Dalam

C++, terdapat berbagai teknik Searching yang dapat digunakan tergantung pada tipe dan organisasi data.

4. Guided

pada guided saya bagi menjadi 2 agar tidak terlalu kecil gambar codingannya dan tidak ngeblur

```
1  #include <iostream>
2
3  using namespace std;
4
5  // Struktur untuk node dalam linked list
6
7  struct Node {
8      int data;
9      Node* next;
10 };
11
12 // Fungsi untuk menambahkan elemen baru ke awal linked list
13 void insertFirst(Node*& head, Node*& tail, int new_data){
14     Node* new_node = new Node();
15     new_node->data = new_data;
16     new_node->next = head;
17     head = new_node;
18
19     if (tail == nullptr) {
20         tail = new_node;
21     }
22 }
23
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
29     if (head == nullptr){
30         head = new_node;
31         tail = new_node;
32     } else {
33         tail->next = new_node;
34         tail = new_node;
35     }
36 }
37
38 int findElement(Node* head, int x){
39     Node* current = head;
40     int index = 0;
41
42     while(current != nullptr){
43         if (current->data == x){
44             return index;
45         }
46         current = current->next;
47         index++;
48     }
49     return -1;
50 }
51
52 void display(Node* node){
53     while (node != nullptr){
54         cout << node->data << " ";
55         node = node->next;
56     }
57     cout << endl;
58 }
```

```
1 void deleteElement(Node*& head, int x){
2     if (head == nullptr){
3         cout << "Linked List kosong" << endl;
4         return;
5     }
6
7     if (head->data == x){
8         Node* temp = head;
9         head = head->next;
10        delete temp;
11        return;
12    }
13
14    Node* current = head;
15    while(current -> next != nullptr){
16        if(current->next->data == x){
17            Node* temp = current->next;
18            current->next = current->next->next;
19            delete temp;
20            return;
21        }
22        current = current->next;
23    }
24 }
25
26 int main()
27 {
28     Node* head = nullptr;
29     Node* tail = nullptr;
30
31     insertFirst(head, tail, 3);
32     insertFirst(head, tail, 5);
33     insertFirst(head, tail, 7);
34
35     insertLast(head, tail, 11);
36     insertLast(head, tail, 14);
37     insertLast(head, tail, 18);
38
39     cout << "Elemen dalam linked list: ";
40     display(head);
41
42     int x;
43     cout << "Masukkan elemen yang ingin dicari: ";
44     cin >> x;
45
46     int result = findElement(head, x);
47
48     if (result == -1)
49         cout << "Elemen tidak ditemukan dalam linked list" << endl;
50     else
51         cout << "Elemen ditemukan pada indeks " << result << endl;
52
53     cout << "Masukkan elemen yang ingin dihapus: ";
54     cin >> x;
55     deleteElement(head,x);
56
57     cout << "Elemen dalam linked list setelah penghapusan: ";
58     display(head);
59
60     return 0;
61 }
```

5. Unguided

1. Codingan dengan file “main(no.1).cpp”

```
1  #include "singlelist(no1).h"
2  #include "singlelist(no.1).cpp"
3  #include <iostream>
4  using namespace std;
5
6  int main() {
7      List L;
8      address P1, P2, P3, P4, P5 = NULL;
9
10     createList(L);
11
12     P1 = alokasi(2);
13     insertFirst(L, P1);
14
15     P2 = alokasi(0);
16     insertFirst(L, P2);
17
18     P3 = alokasi(8);
19     insertFirst(L, P3);
20
21     P4 = alokasi(12);
22     insertFirst(L, P4);
23
24     P5 = alokasi(9);
25     insertFirst(L, P5);
26
27     printInfo(L);
28
29     return 0;
30 }
```

Codingan dengan file “singlelist(no.1).cpp”

```
1  #include <iostream>
2  #include "singlelist(no1).h"
3
4  using namespace std;
5
6  void createList(List &L) {
7      L.First = NULL;
8  }
9
10 address alokasi(infotype x) {
11     address P = new Elmlist;
12     if (P != NULL) {
13         P->info = x;
14         P->next = NULL;
15     }
16     return P;
17 }
18
19 void dealokasi(address &P) {
20     delete P;
21     P = NULL;
22 }
23
24 void printInfo(const List &L) {
25     address P = L.First;
26     while (P != NULL) {
27         cout << P->info << " ";
28         P = P->next;
29     }
30     cout << endl;
31 }
32
33 void insertFirst(List &L, address P) {
34     P->next = L.First;
35     L.First = P;
36 }
37
```

Codingan dengan file “singlelist(no.1).h”

```
1  #ifndef SINGLELIST_H
2  #define SINGLELIST_H
3
4  typedef int infotype;
5  typedef struct Elmlist *address;
6
7  struct Elmlist {
8      infotype info;
9      address next;
10 };
11
12 struct List {
13     address First;
14 };
15
16 void createList(List &L);
17 address alokasi(infotype x);
18 void dealokasi(address &P);
19 void printInfo(const List &L);
20 void insertFirst(List &L, address P);
21
22 #endif
23
```

Hasil outputannya:

```
9 12 8 0 2
PS D:\STRUKTUR DATA P5>
```

2. Codingan dengan file “main(no.2).cpp”

```
1  #include <iostream>
2  #include "singlelist(no.2).h"
3  #include "singlelist(no.2).cpp"
4
5  using namespace std;
6
7  int main() {
8      List L;
9      address P1, P2, P3, P4, P5 = NULL;
10
11      CreateList(L);
12
13      P1 = alokasi(2);
14      insertFirst(L, P1);
15
16      P2 = alokasi(0);
17      insertFirst(L, P2);
18
19      P3 = alokasi(8);
20      insertFirst(L, P3);
21
22      P4 = alokasi(12);
23      insertFirst(L, P4);
24
25      P5 = alokasi(9);
26      insertFirst(L, P5);
27
28      address foundElement = findElm(L, 8);
29      if (foundElement != NULL) {
30          cout << "8 ditemukan dalam list" << endl;
31      } else {
32          cout << "8 tidak ditemukan dalam list" << endl;
33      }
34
35      return 0;
36 }
```

Codingan dengan file “singlelist(no.2).cpp”

```

1  #include "singlelist(no.2).h"
2  #include <iostream>
3
4  void Createlist(List &L) {
5      L.First = NULL;
6  }
7
8  address alokasi(infotype x) {
9      address P = new Elmlist;
10     P->info = x;
11     P->next = NULL;
12     return P;
13 }
14
15 void dealokasi(address &P) {
16     delete P;
17 }
18
19 void printInfo(List L) {
20     address P = L.First;
21     while (P != NULL) {
22         std::cout << P->info << " ";
23         P = P->next;
24     }
25     std::cout << std::endl;
26 }
27
28 void insertFirst(List &L, address P) {
29     P->next = L.First;
30     L.First = P;
31 }
32
33 address findElm(List L, infotype x) {
34     address P = L.First;
35     while (P != NULL) {
36         if (P->info == x) {
37             return P;
38         }
39         P = P->next;
40     }
41     return NULL;
42 }
43
44 int sumInfo(List L) {
45     int total = 0;
46     address P = L.First;
47     while (P != NULL) {
48         total += P->info;
49         P = P->next;
50     }
51     return total;
52 }

```

Codingan dengan file “singlelist(no.2).h”

```

1  #ifndef SINGLELIST_H
2  #define SINGLELIST_H
3
4  #include <iostream>
5
6  typedef int infotype;
7  typedef struct Elmlist *address;
8
9  struct Elmlist {
10     infotype info;
11     address next;
12 };
13
14 struct List {
15     address First;
16 };
17
18 void Createlist(List &L);
19 address alokasi(infotype x);
20 void dealokasi(address &P);
21 void printInfo(List L);
22 void insertFirst(List &L, address P);
23 address findElm(List L, infotype x);
24 int sumInfo(List L);
25
26 #endif

```


Outputannya:

```
8 ditemukan dalam list
PS D:\STRUKTUR DATA P5> █
```

3. Codengan dengan file “main(no.3).cpp”

```
1  #include <iostream>
2  #include "singlelist(no.3).h"
3  #include "singlelist(no.3).cpp"
4
5  using namespace std;
6
7  int main() {
8      List L;
9      address P1, P2, P3, P4, P5 = NULL;
10
11      // Membuat list kosong
12      createList(L);
13
14      // Menambahkan elemen ke list
15      P1 = alokasi(2);
16      insertFirst(L, P1);
17
18      P2 = alokasi(0);
19      insertFirst(L, P2);
20
21      P3 = alokasi(8);
22      insertFirst(L, P3);
23
24      P4 = alokasi(12);
25      insertFirst(L, P4);
26
27      P5 = alokasi(9);
28      insertFirst(L, P5);
29
30      // Menampilkan elemen dalam list
31      printInfo(L);
32
33      // Menghitung total nilai elemen
34      int total = sumInfo(L);
35      cout << "Total info dari kelima elemen adalah " << total << endl;
36
37      return 0;
38  }
39
```

Codingan dengan file “singlelist(no.3).cpp”

```
1  #include <iostream>
2  #include "singlelist(no.3).h"
3
4
5  using namespace std;
6
7  // Membuat list kosong
8  void createlist(List &L) {
9      L.First = NULL;
10 }
11
12 // Mengalokasikan elemen baru
13 address alokasi(infotype x) {
14     address P = new Elmlist;
15     if (P != NULL) {
16         P->info = x;
17         P->next = NULL;
18     }
19     return P;
20 }
21
22 // Dealokasi atau menghapus elemen
23 void dealokasi(address &P) {
24     delete P;
25     P = NULL;
26 }
27
28 // Menampilkan informasi elemen di dalam list
29 void printInfo(const List &L) {
30     address P = L.First;
31     while (P != NULL) {
32         cout << P->info << " ";
33         P = P->next;
34     }
35     cout << endl;
36 }
37
38 // Menambahkan elemen di awal list
39 void insertFirst(List &L, address P) {
40     P->next = L.First;
41     L.First = P;
42 }
43
44 // Menjumlahkan seluruh elemen dalam list
45 int sumInfo(const List &L) {
46     address P = L.First;
47     int total = 0;
48     while (P != NULL) {
49         total += P->info; // Menambahkan nilai elemen
50         P = P->next;
51     }
52     return total; // Mengembalikan hasil penjumlahan
53 }
54
```

Codingan dengan file “singlelist(no.3).h

```
1  #ifndef SINGLELIST_H
2  #define SINGLELIST_H
3
4  typedef int infotype;
5  typedef struct Elmlist *address;
6
7  struct Elmlist {
8      infotype info;
9      address next;
10 };
11
12 struct List {
13     address First;
14 };
15
16 // Prosedur dan Fungsi ADT
17 void createlist(List &L);
18 address alokasi(infotype x);
19 void dealokasi(address &P);
20 void printInfo(const List &L);
21 void insertFirst(List &L, address P);
22 int sumInfo(const List &L); // Fungsi untuk menjumlahkan elemen-elemen
23
24 #endif
25
```

Outputannya:

```
9 12 8 0 2  
Total info dari kelima elemen adalah 31  
PS D:\STRUKTUR DATA P5>
```

6. Kesimpulan

Kesimpulannya yaitu Searching adalah operasi penting dalam ilmu komputer yang digunakan untuk menemukan elemen tertentu dalam struktur data. Dalam C++, terdapat berbagai teknik pencarian yang bisa digunakan sesuai dengan tipe dan organisasi data yang sedang diolah.