

Tugas Pendahuluan Modul 4
STRUKTUR DATA - Genap 2024/2025
"Single Linked List "

A. Ketentuan Tugas Pendahuluan

1. Tugas Pendahuluan dikerjakan secara **Individu**.
2. TP ini bersifat **WAJIB**, tidak mengerjakan = **PENGURANGAN POIN JURNAL / TES ASESMEN**.
3. Hanya **MENGUMPULKAN** tetapi **TIDAK MENGERJAKAN** = **PENGURANGAN POIN JURNAL / TES ASESMEN**.
4. Deadline pengumpulan TP Modul 4 adalah Senin, 9 Oktober 2023 pukul 06.00 WIB.
5. **TIDAK ADA TOLERANSI KETERLAMBATAN, TERLAMBAT ATAU TIDAK MENGUMPULKAN TP MAKA DIANGGAP TIDAK MENGERJAKAN**.
6. **DILARANG PLAGIAT (PLAGIAT = E)**.
7. Kerjakan TP dengan jelas agar dapat dimengerti.
8. File diupload di LMS menggunakan format **PDF** dengan ketentuan:
TP_MOD_[XX]_NIM_NAMA.pdf
9. **SOAL TEORI WAJIB DIKERJAKAN TULIS TANGAN, TIDAK BOLEH DIKETIK!**

CP (WA):

- Raihan (089638482851)
- Kayyisa (085105303555)
- Abiya (082127180662)
- Rio (081210978384)

SELAMAT MENGERJAKAN^^

LAPORAN PRAKTIKUM
PERTEMUAN 4
SINGLE LINKED LIST BAGIAN PERTAMA



Nama :

Tsaqif Hisyam Saputra (2311104024)

Dosen :

Yudha Islami Sulistiya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

I. TUJUAN

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

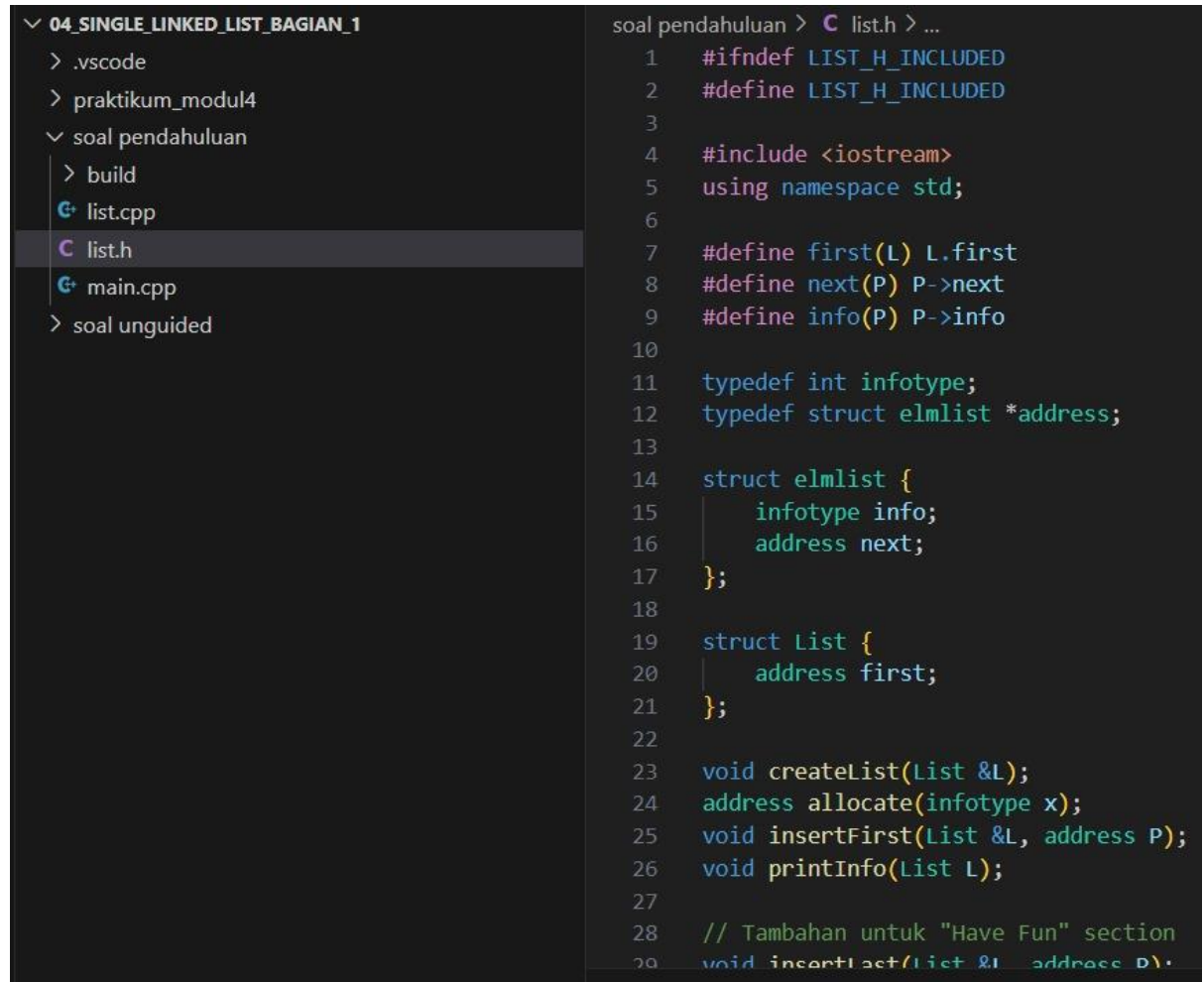
II. TOOL

Visual Studio Code dengan C++ Extensions

III. UNGUIDED

Soal Pendahuluan

list.h



```
04_SINGLE_LINKED_LIST_BAGIAN_1
├── .vscode
├── praktikum_modul4
├── soal_pendahuluan
│   ├── build
│   ├── list.cpp
│   └── list.h
└── main.cpp
└── soal_unguided

soal_pendahuluan > C list.h > ...
1  #ifndef LIST_H_INCLUDED
2  #define LIST_H_INCLUDED
3
4  #include <iostream>
5  using namespace std;
6
7  #define first(L) L.first
8  #define next(P) P->next
9  #define info(P) P->info
10
11 typedef int infotype;
12 typedef struct elmlist *address;
13
14 struct elmlist {
15     infotype info;
16     address next;
17 };
18
19 struct List {
20     address first;
21 };
22
23 void createList(List &L);
24 address allocate(infotype x);
25 void insertFirst(List &L, address P);
26 void printInfo(List L);
27
28 // Tambahan untuk "Have Fun" section
29 void insertLast(List &L, address P);
```

File ini merupakan header file yang mendeklarasikan struktur data dan fungsi-fungsi yang akan digunakan dalam program. Berikut adalah komponen yang terdapat dalam file:

Struktur data:

- **'elmlist'**: Struktur yang mendefinisikan elemen dari linked list, yang terdiri dari informasi (**'info'**) dan pointer ke elemen berikutnya (**'next'**).
- **'List'**: Struktur yang mendefinisikan linked list itu sendiri, yang memiliki pointer **'first'** yang menunjuk ke elemen pertama dalam list.

Deklarasi Fungsi:

- **createList(List &L)**: Untuk menginisialisasi list kosong.
- **address allocate(infotype x)**: Untuk mengalokasikan memori untuk elemen baru dan mengisi informasinya.
- **void insertFirst(List &L, address P)**: Untuk menyisipkan elemen baru di awal list.
- **void printInfo(List L)**: Untuk mencetak semua elemen dalam list.

list.cpp

```
04_SINGLE_LINKED_LIST_BAGIAN_1
> .vscode
> praktikum_modul4
> soal pendahuluan
  > build
  list.cpp
  list.h
  main.cpp
  > soal unguided

soal pendahuluan > list.cpp > searchInfo(List, infotype)
1  #include "list.h"
2
3  void createList(List &L) {
4      first(L) = NULL;
5  }
6
7  address allocate(infotype x) {
8      address P = new elmlist;
9      info(P) = x;
10     next(P) = NULL;
11     return P;
12 }
13
14 void insertFirst(List &L, address P) {
15     next(P) = first(L);
16     first(L) = P;
17 }
18
19 void printInfo(List L) {
20     address P = first(L);
21     while (P != NULL) {
22         cout << info(P) << " ";
23         P = next(P);
24     }
25     cout << endl;
26 }
27
28 // Implementasi fungsi tambahan untuk "Have Fun" section
29 void insertLast(List &L, address P) {
```

File ini berisi implementasi dari fungsi-fungsi yang telah dideklarasikan dalam **list.h**. Setiap fungsi di sini memiliki implementasi yang mendetail sesuai dengan deskripsi yang telah diberikan sebelumnya. Lalu di bawah ini adalah beberapa fungsinya:

Fungsi implementasi:

- **createList:** Mengatur list menjadi kosong.
- **allocate:** Mengalokasi memori untuk elemen baru dan mengisinya dengan nilai yang diberikan.
- **insertFirst:** Menyisipkan elemen baru di awal list.
- **printInfo:** Mencetak semua elemen dalam list.
- **InsertLast:** Menyisipkan elemen baru di akhir list.
- **deleteLast:** Menghapus elemen terakhir dari list.
- **deleteAfter:** Menghapus elemen setelah elemen tertentu.
- **searchInfo:** Mencari elemen dalam list berdasarkan nilai yang diberikan.

main.cpp

```
04_SINGLE_LINKED_LIST_BAGIAN_1
├── .vscode
├── praktikum_modul4
├── soal_pendahuluan
│   ├── build
│   ├── list.cpp
│   ├── list.h
│   └── main.cpp
└── soal_unguided

soal_pendahuluan > main.cpp > main()
1  #include "list.h"
2
3  int main() {
4      List L;
5      createList(L);
6
7      cout << "Masukkan NIM perdigit" << endl;
8      for (int i = 1; i <= 10; i++) {
9          int digit;
10         cout << "Digit " << i << " : ";
11         cin >> digit;
12         address P = allocate(digit);
13         insertLast(L, P);
14     }
15 }
16
```

File ini berfungsi sebagai titik masuk program. Di dalamnya dilakukan hal seperti ini:

- Membuat List: Menginisialisasi list dengan memanggil **createList**.
- Input Pengguna: Meminta pengguna untuk memasukan 10 digit NIM, satu per satu. Setiap digit akan dialokasikan sebagai elemen baru dan disisipkan di akhir list menggunakan **insertLast**.
- Output: Meskipun tidak ada fungsi mencetak isi list yang ditambahkan dalam kode ini, Dapat ditambahkan panggilan ke **printInfo** setelah proses input untuk menampilkan list.

```
* Executing task: C:/Windows/System32/cmd.exe /d /c .\build\Debug\outDebug.exe

Masukkan nomor NIM ke-1: 3
List setelah memasukkan data ke-1: 3
Masukkan nomor NIM ke-2: 4
List setelah memasukkan data ke-2: 4 3
Masukkan nomor NIM ke-3: 5
List setelah memasukkan data terakhir: 5 4 3
* Terminal will be reused by tasks, press any key to close it.
```

Latihan Soal

1.Create SLL

```
SOAL UNGUIDED
> .vscode
> build
createSLL.cpp
deletenode.cpp
searchSLL.cpp

createSLL.cpp > ...
11 struct LinkedList {
28     void insertBack(int value) {
45
46     // Fungsi untuk mencetak seluruh isi linked list
47     void printList() {
48         Node* temp = head;
49         while (temp != nullptr) {
50             cout << temp->data;
51             if (temp->next != nullptr) {
52                 cout << " -> "; // Memisahkan elemen dengan "->"
53             }
54             temp = temp->next;
55         }
56         cout << endl;
57     }
58 };
59
60 // Program utama
61 int main() {
62     LinkedList list; // Membuat linked list kosong
63
64     // Input seperti pada soal
65     list.insertFront(20);
66     list.insertBack(30);
67     list.insertFront(15);
68
69     // Cetak linked list
70     cout << "Isi Linked List: ";
71     list.printList();
72
73     return 0;
74 }
75
```

Struct **Node**: Setiap elemen dari linked list diwakili oleh **Node**. Setiap node menyimpan data (nilai **int**) dan pointer **next** yang menunjuk ke node berikutnya.

Struct **LinkedList**:

- **head** adalah pointer yang menunjuk ke elemen pertama dari linked list.
- **insertFront(int value)**: Menambahkan node baru di depan linked list. Node baru akan menjadi elemen pertama (**head**).
- **insertBack(int value)**: Menambahkan node baru di belakang linked list. Program mencari node terakhir dan menghubungkan node baru setelahnya.
- **printList()**: Mencetak seluruh linked list, dengan format elemen yang dipisahkan oleh **->**.

Main():

- Program menambahkan node dengan nilai 20 di depan, kemudian node dengan nilai 30 di belakang, dan nilai node 15 di depan.

Hasilnya akan mencetak :

```
Isi Linked List: 15 -> 20 -> 30
```

```
PS C:\DEVELOPMENT\C++\Struktur Data\04_Single_Linked_List_Bagian_1\soal unguided>
```

2.Delete Node

```
SOAL UNGUIDED
> .vscode
> build
createSLL.cpp
deletenode.cpp
deletenode.exe
searchSLL.cpp
searchSLL.exe

deletenode.cpp > main()
11 struct LinkedList {
79 void printList() {
88     cout << endl;
89 }
90 };
91
92 // Program utama
93 int main() {
94     LinkedList list; // Membuat linked list kosong
95
96     // Input seperti pada soal
97     list.insertFront(15);
98     list.insertBack(25);
99     list.insertFront(10);
100
101     // Cetak linked list sebelum penghapusan
102     cout << "Isi Linked List sebelum penghapusan: ";
103     list.printList();
104
105     // Menghapus node dengan nilai 10
106     list.deleteNode(10);
107
108     // Cetak linked list setelah penghapusan
109     cout << "Isi Linked List setelah penghapusan: ";
110     list.printList();
111
112     return 0;
113 }
```

deleteNode(int value): Fungsi ini bertanggung jawab untuk menghapus node yang memiliki nilai tertentu:

- Jika list kosong, program akan memberi tahu bahwa tidak ada yang bisa dihapus.
- Jika node pertama (head) yang memiliki nilai yang akan dihapus, program akan menghapus node tersebut dan memindahkan pointer **head** ke node berikutnya.
- Jika node yang akan dihapus berada di tengah atau akhir, program akan mencari node tersebut dan menghapusnya dari list.

printList(): Fungsi ini digunakan untuk mencetak isi dari linked list.

Main(): Program menambahkan beberapa node (dengan nilai 10->15->25) ke dalam linked list, lalu menghapus node dengan nilai 10, dan kemudian mencetak isi linked list sebelum dan sesudah penghapusan.

```
Isi Linked List sebelum penghapusan: 10 -> 15 -> 25
Isi Linked List setelah penghapusan: 15 -> 25
PS C:\DEVELOPMENT\C++\Struktur Data\04_Single_Linked_List_Bagian_1\soal unguided>
```


3.Search SLL

```
SOAL UNGUIDED
> .vscode
> build
createSLL.cpp
deletenode.cpp
searchSLL.cpp

searchSLL.cpp > ...
83 // Program utama
84 int main() {
85     LinkedList list; // Membuat linked list kosong
86
87     // Input seperti pada soal
88     list.insertFront(20);
89     list.insertBack(40);
90     list.insertFront(10);
91
92     // Cetak linked list
93     cout << "Isi Linked List: ";
94     list.printList();
95
96     // Mencari node dengan nilai 40
97     if (list.searchNode(40)) {
98         cout << "Node dengan nilai 40 ditemukan." << endl;
99     } else {
100         cout << "Node dengan nilai 40 tidak ditemukan." << endl;
101     }
102
103     // Menghitung panjang linked list
104     cout << "Panjang linked list: " << list.countNodes() << endl; // Output: 3
105
106     return 0;
107 }
108
```

searchNode(int value): Fungsi ini digunakan untuk mencari node dengan nilai tertentu di dalam linked list:

- Fungsi akan melakukan penelusuran dari node pertama hingga akhir.
- Jika menemukan node dengan nilai yang dicari, fungsi mengembalikan **true**; jika tidak, mengembalikan **false**.

countNodes(): Fungsi ini menghitung jumlah node di dalam linked list:

- Fungsi akan melakukan penelusuran seluruh node dalam linked list, menambahkan satu ke dalam counter setiap kali node ditemukan.

printList(): Fungsi ini mencetak isi dari linked list.

main():

- Program menambahkan beberapa node (dengan nilai 10, 20, 40) ke dalam linked list.
- Mencari node dengan nilai 20 dan mencetak hasil pencarian.
- Menghitung jumlah node dalam linked list dan mencetak hasilnya.

Output Program:

```
Isi Linked List: 10 -> 20 -> 40
Node dengan nilai 40 ditemukan.
Panjang linked list: 3
PS C:\DEVELOPMENT\C++\Struktur Data\04_Single_Linked_List_Bagian_1\soal unguided>
```

IV. GUIDED

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 // Deklarasi Struct untuk mahasiswa
6 struct mahasiswa {
7     char nama[50];
8     char nim[20];
9 };
10
11 // Deklarasi Struct Node
12 struct Node {
13     mahasiswa data;
14     Node *next;
15 };
16
17 Node *head;
18 Node *tail;
19
20 // Inisialisasi list
21 void Init() {
22     head = nullptr;
23     tail = nullptr;
24 }
25
26 // Mengecek apakah list kosong
27 bool isEmpty() {
28     return head == nullptr;
29 }
30
31 // Tambah Depan
32 void insertDepan(const mahasiswa &data) {
33     Node *baru = new Node;
34     baru->data = data;
35     baru->next = head;
36     if (isEmpty()) {
37         head = tail = baru;
38     } else {
39         baru->next = head;
40         head = baru;
41     }
42 }
43
44 // Tambah Belakang
45 void insertBelakang(const mahasiswa &data) {
46     Node *baru = new Node;
47     baru->data = data;
48     baru->next = nullptr;
49     if (isEmpty()) {
50         head = tail = baru;
51     } else {
52         tail->next = baru;
53         tail = baru;
54     }
55 }
56
57 // Hitung Jumlah list
58 int hitungList() {
59     Node *current = head;
60     int jumlah = 0;
61     while (current != nullptr) {
62         jumlah++;
63         current = current->next;
64     }
65     return jumlah;
66 }
67
68 // Hapus Depan
69 void hapusDepan() {
70     if (!isEmpty()) {
71         Node *hapus = head;
72         head = head->next;
73         delete hapus;
74         if (head == nullptr) {
75             tail = nullptr; // jika list menjadi kosong
76         }
77     } else {
78         cout << "List kosong!" << endl;
79     }
80 }
81
82 // Hapus Belakang
83 void hapusBelakang() {
84     if (!isEmpty()) {
85         if (head == tail) {
86             delete head;
87             head = tail = nullptr; // list menjadi kosong
88         } else {
89             Node *bantu = head;
90             while (bantu->next != tail) {
91                 bantu = bantu->next;
92             }
93             delete tail;
94             tail = bantu;
95             bantu->next = nullptr;
96         }
97     } else {
98         cout << "List kosong!" << endl;
99     }
100 }
101
102 // Tampilkan list
103 void tampil() {
104     Node *current = head;
105     if (!isEmpty()) {
106         while (current != nullptr) {
107             cout << "Nama: " << current->data.nama << ", NIM: " << current->data.nim << endl;
108             current = current->next;
109         }
110     } else {
111         cout << "List masih kosong!" << endl;
112     }
113 }
114
115 // Hapus list
116 void clearList() {
117     Node *current = head;
118     while (current != nullptr) {
119         Node *hapus = current;
120         current = current->next;
121         delete hapus;
122     }
123     head = tail = nullptr;
124     cout << "List berhasil terhapus!" << endl;
125 }
126
127 // Main function
128 int main() {
129     Init();
130
131     // Contoh data mahasiswa
132     mahasiswa m1 = {"Ali", "123456"};
133     mahasiswa m2 = {"Budi", "654321"};
134     mahasiswa m3 = {"Charlie", "112233"};
135
136     // Menambahkan mahasiswa ke dalam list
137     insertDepan(m1);
138     tampil();
139     insertBelakang(m2);
140     tampil();
141     insertDepan(m3);
142     tampil();
143
144     // Menghapus elemen dari list
145     hapusDepan();
146     tampil();
147     hapusBelakang();
148     tampil();
149
150     // Menghapus seluruh list
151     clearList();
152     tampil();
153     return 0;
154 }
```

```

1 #include <iostream>
2 using namespace std;
3
4 // Definisi struktur untuk elemen list
5 struct Node {
6     int data;           // Menyimpan nilai elemen
7     Node* next;        // Pointer ke elemen berikutnya
8 };
9
10 // Fungsi untuk mengalokasikan memori untuk node baru
11 Node* alokasi(int value) {
12     Node* newNode = new Node; // Alokasi memori untuk elemen baru
13     if (newNode != nullptr) { // Jika alokasi berhasil
14         newNode->data = value; // Mengisi data node
15         newNode->next = nullptr; // Set next ke nullptr
16     }
17     return newNode; // Mengembalikan pointer node baru
18 }
19
20 // Fungsi untuk dealokasi memori node
21 void dealokasi(Node* node) {
22     delete node; // Mengembalikan memori yang digunakan oleh node
23 }
24
25 // Pengecekan apakah list kosong
26 bool isEmpty(Node* head) {
27     return head == nullptr; // List kosong jika head adalah nullptr
28 }
29
30 // Menambahkan elemen di awal list
31 void insertFirst(Node* &head, int value) {
32     Node* newNode = alokasi(value); // Alokasi memori untuk elemen baru
33     if (newNode != nullptr) {
34         newNode->next = head; // Menghubungkan elemen baru ke elemen pertama
35         head = newNode;      // Menetapkan elemen baru sebagai elemen pertama
36     }
37 }
38
39 // Menambahkan elemen di akhir list
40 void insertLast(Node* &head, int value) {
41     Node* newNode = alokasi(value); // Alokasi memori untuk elemen baru
42     if (newNode != nullptr) {
43         if (isEmpty(head)) { // Jika list kosong
44             head = newNode; // Elemen baru menjadi elemen pertama
45         } else {
46             Node* temp = head;
47             while (temp->next != nullptr) { // Mencari elemen terakhir
48                 temp = temp->next;
49             }
50             temp->next = newNode; // Menambahkan elemen baru di akhir list
51         }
52     }
53 }
54
55 // Menampilkan semua elemen dalam list
56 void printList(Node* head) {
57     if (isEmpty(head)) {
58         cout << "List kosong!" << endl;
59     } else {
60         Node* temp = head;
61         while (temp != nullptr) { // Selama belum mencapai akhir list
62             cout << temp->data << " "; // Menampilkan data elemen
63             temp = temp->next; // Melanjutkan ke elemen berikutnya
64         }
65         cout << endl;
66     }
67 }
68
69 // Menghitung jumlah elemen dalam list
70 int countElements(Node* head) {
71     int count = 0;
72     Node* temp = head;
73     while (temp != nullptr) {
74         count++; // Menambah jumlah elemen
75         temp = temp->next; // Melanjutkan ke elemen berikutnya
76     }
77     return count; // Mengembalikan jumlah elemen
78 }
79
80 // Menghapus semua elemen dalam list dan dealokasi memori
81 void clearList(Node* &head) {
82     while (head != nullptr) {
83         Node* temp = head; // Simpan pointer ke node saat ini
84         head = head->next; // Pindahkan ke node berikutnya
85         dealokasi(temp); // Dealokasi node
86     }
87 }
88
89 int main() {
90     Node* head = nullptr; // Membuat list kosong
91
92     // Menambahkan elemen ke dalam list
93     insertFirst(head, 10); // Menambahkan elemen 10 di awal list
94     insertLast(head, 20); // Menambahkan elemen 20 di akhir list
95     insertLast(head, 30); // Menambahkan elemen 30 di akhir list
96
97     // Menampilkan isi list
98     cout << "Isi List: ";
99     printList(head);
100
101     // Menampilkan jumlah elemen
102     cout << "Jumlah elemen: " << countElements(head) << endl;
103
104     // Menghapus semua elemen dalam list
105     clearList(head);
106
107     // Menampilkan isi list setelah penghapusan
108     cout << "Isi List setelah penghapusan: ";
109     printList(head);
110
111     return 0;
112 }

```