

**LAPORAN PRAKTIKUM**  
**Tugas Pendahuluan Modul 05**  
**“Single Linked List”**



**Disusun Oleh:**  
**Hamidatun Nisa - 21104063**  
**Struktur Data SE07-01**

**Dosen :**  
**Yudha Islami Sulistya**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**  
**FAKULTAS INFORMATIKA**  
**UNIVERSITAS TELKOM PURWOKERTO**  
**2024**

## 1. Mencari Elemen Tertentu dalam SLL

Code :

```
01 Single_Linked_List_Bagian_2 > TP > @ lathan_01.cpp > ...
02 #include <iostream>
03 #include <string>
04 using namespace std;
05
06 struct Node {
07     int data;
08     Node* next;
09     Node* prev;
10 };
11
12 struct list {
13     Node* head;
14     Node* tail;
15 };
16
17 void createlist_21104063(list &l) {
18     l.head = nullptr;
19     l.tail = nullptr;
20 }
21
22 Node* alokasi_21104063(int value) {
23     Node* newnode = new Node;
24     newnode->data = value;
25     newnode->next = nullptr;
26     newnode->prev = nullptr;
27     return newnode;
28 }
29
30 void insertfirst_21104063(list &l, Node* P) {
31     if (l.head == nullptr) {
32         l.head = P;
33         l.tail = P;
34     } else {
35         P->next = l.head;
36         l.head->prev = P;
37         l.head = P;
38     }
39 }
40
41 void insertlast_21104063(list &l, Node* P) {
42     if (l.tail == nullptr) {
43         l.head = P;
44         l.tail = P;
45     } else {
46         P->prev = l.tail;
47         l.tail->next = P;
48         l.tail = P;
49 }
50
51 void deletefirst_21104063(list &l) {
52     if (l.head == nullptr) {
53         cout << "List kosong, tidak ada elemen yang bisa dihapus";
54     } else if (l.head == l.tail) {
55         delete l.head;
56         l.head = nullptr;
57         l.tail = nullptr;
58     } else {
59         Node* temp = l.head;
60         l.head = l.head->next;
61         l.head->prev = nullptr;
62         delete temp;
63     }
64 }
65
66 void deletelast_21104063(list &l) {
67     if (l.tail == nullptr) {
68         cout << "List kosong, tidak ada elemen yang bisa dihapus";
69     } else if (l.head == l.tail) {
70         delete l.tail;
71         l.head = nullptr;
72         l.tail = nullptr;
73     } else {
74         Node* temp = l.tail;
75         l.tail = l.tail->prev;
76         l.tail->next = nullptr;
77         delete temp;
78     }
79 }
80
81 void bubbleSortlist_21104063(list &l) {
82     if (l.head == nullptr || l.head->next == nullptr) {
83         return;
84     }
85     bool swapped;
86     do {
87         swapped = false;
88         Node* current = l.head;
89         while (current->next != nullptr) {
90             if (current->data > current->next->data) {
91                 int temp = current->data;
92                 current->data = current->next->data;
93                 current->next->data = temp;
94                 swapped = true;
95             }
96             current = current->next;
97         }
98     } while (swapped);
99 }
100
101 void printlistfromhead_21104063(list l) {
102     if (l.head == nullptr) {
103         cout << endl;
104     } else {
105         Node* current = l.head;
106         while (current != nullptr) {
107             cout << "Masukkan elemen ke-" << l << " = ";
108             cin >> value;
109             Node* newnode = alokasi_21104063(value);
110             insertlast_21104063(l, newnode);
111             current = current->next;
112         }
113     }
114 }
115
116 void searchElement_21104063(list l, int value) {
117     Node* current = l.head;
118     int position = 1;
119     bool found = false;
120     while (current != nullptr) {
121         if (current->data == value) {
122             cout << "Elemen ditemukan pada posisi ke-" << position << endl;
123             found = true;
124             break;
125         }
126         current = current->next;
127         position++;
128     }
129     if (!found) {
130         cout << "Elemen dengan nilai " << value << " tidak ditemukan";
131     }
132 }
133
134 int main() {
135     list l;
136     createlist_21104063(l);
137     inputElements_21104063(l);
138     int cari;
139     cout << "Masukkan nilai yang ingin dicari : ";
140     cin >> cari;
141     searchElement_21104063(l, cari);
142     return 0;
143 }
```

Output :

```
Masukkan elemen ke-1: 30
Masukkan elemen ke-2: 45
Masukkan elemen ke-3: 21
Masukkan elemen ke-4: 40
Masukkan elemen ke-5: 89
Masukkan elemen ke-6: 76

Masukkan nilai yang ingin dicari: 76
Elemen ditemukan pada posisi ke-6 dengan alamat: 0x73edd0
PS D:\= Collage\Semester 7\Praktikum Struktur Data\STD_Hamidatun_Nisa_21104063>
```

```
Masukkan elemen ke-1: 60
Masukkan elemen ke-2: 32
Masukkan elemen ke-3: 56
Masukkan elemen ke-4: 86
Masukkan elemen ke-5: 46
Masukkan elemen ke-6: 33

Masukkan nilai yang ingin dicari: 10
Elemen dengan nilai 10 tidak ditemukan dalam list.
PS D:\= Collage\Semester 7\Praktikum Struktur Data\STD_Hamidatun_Nisa_21104063>
```

## 2. Mengurutkan List Menggunakan Bubble Sort

Code :

```
05 Single_Linked_List_Bagian_2 > TP > @ lathan_02.cpp > ...
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 struct Node {
7     int data;
8     Node* next;
9     Node* prev;
10 };
11
12 struct List {
13     Node* head;
14     Node* tail;
15 };
16
17 void createList_21104063(List &l) {
18     l.head = nullptr;
19     l.tail = nullptr;
20 }
21
22 Node* alokasi_21104063(int value) {
23     Node* newNode = new Node;
24     newNode->data = value;
25     newNode->next = nullptr;
26     newNode->prev = nullptr;
27     return newNode;
28 }
29
30 void insertFirst_21104063(List &l, Node* P) {
31     if (l.head == nullptr) {
32         l.head = P;
33         l.tail = P;
34     } else {
35         P->next = l.head;
36         l.head->prev = P;
37         l.head = P;
38     }
39 }
40
41 void insertLast_21104063(List &l, Node* P) {
42     if (l.tail == nullptr) {
43         l.head = P;
44         l.tail = P;
45     } else {
46         P->prev = l.tail;
47         l.tail->next = P;
48         l.tail = P;
49     }
50 }
51
52 void deleteFirst_21104063(List &l) {
53     if (l.head == nullptr) {
54         cout << "list kosong, tidak ada elemen yang bisa dihapus";
55     } else if (l.head == l.tail) {
56         delete l.head;
57         l.head = nullptr;
58         l.tail = nullptr;
59     } else {
60         Node* temp = l.head;
61         l.head = l.head->next;
62         l.head->prev = nullptr;
63         delete temp;
64     }
65 }
66
67 void deleteLast_21104063(List &l) {
68     if (l.tail == nullptr) {
69         cout << "list kosong, tidak ada elemen yang bisa dihapus";
70     } else if (l.head == l.tail) {
71         delete l.tail;
72         l.head = nullptr;
73         l.tail = nullptr;
74     } else {
75         Node* temp = l.tail;
76         l.tail = l.tail->prev;
77         l.tail->next = nullptr;
78         delete temp;
79     }
80 }
81
82 void bubbleSortList_21104063(List &l) {
83     if (l.head == nullptr || l.head->next == nullptr) {
84         return;
85     }
86     bool swapped;
87     do {
88         swapped = false;
89         Node* current = l.head;
90         while (current->next != nullptr) {
91             if (current->data > current->next->data) {
92                 int temp = current->data;
93                 current->data = current->next->data;
94                 current->next->data = temp;
95                 swapped = true;
96             }
97             current = current->next;
98         }
99     } while (swapped);
100 }
101
102 void printListFromHead_21104063(List l) {
103     if (l.head == nullptr) {
104         cout << "list kosong." << endl;
105     } else {
106         Node* temp = l.head;
107         while (temp != nullptr) {
108             cout << temp->data;
109             if (temp->next != nullptr) {
110                 cout << " <-> ";
111             }
112             temp = temp->next;
113         }
114         cout << endl;
115     }
116 }
117
118 void inputElements_21104063(List &l) {
119     for (int i = 1; i <= 5; i++) {
120         int value;
121         cout << "Masukkan elemen ke-" << i << " : ";
122         cin >> value;
123         Node* newNode = alokasi_21104063(value);
124         insertLast_21104063(l, newNode);
125     }
126 }
127
128 int main() {
129     List l;
130     createList_21104063(l);
131     inputElements_21104063(l);
132     cout << "List sebelum diurutkan: ";
133     printListFromHead_21104063(l);
134     bubbleSortList_21104063(l);
135     cout << "List setelah diurutkan: ";
136     printListFromHead_21104063(l);
137     return 0;
138 }
```

Output :

```
Masukkan elemen ke-1 = 54
Masukkan elemen ke-2 = 32
Masukkan elemen ke-3 = 67
Masukkan elemen ke-4 = 53
Masukkan elemen ke-5 = 66

List sebelum diurutkan: 54 <-> 32 <-> 67 <-> 53 <-> 66

List setelah diurutkan: 32 <-> 53 <-> 54 <-> 66 <-> 67
PS D:\= Collage\Semester 7\Praktikum Struktur Data\STD_Hamidatun_Nisa_21104063>
```

### 3. Menambahkan Elemen Secara Terurut

Code :

```
05 Single_Linked_List_Bagian_2 > TP > @ lshah_03.cpp > ...
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 struct Node {
7     int data;
8     Node* next;
9     Node* prev;
10 };
11
12 struct list {
13     Node* head;
14     Node* tail;
15 };
16
17 void createlist_21104063(list &l) {
18     l.head = nullptr;
19     l.tail = nullptr;
20 }
21
22 Node* alokasi_21104063(int value) {
23     Node* newNode = new Node;
24     newNode->data = value;
25     newNode->next = nullptr;
26     newNode->prev = nullptr;
27     return newNode;
28 }
29
30 void insertFirst_21104063(list &l, Node* P) {
31     if (l.head == nullptr) {
32         l.head = P;
33         l.tail = P;
34     } else {
35         P->next = l.head;
36         l.head->prev = P;
37         l.head = P;
38     }
39 }
40
41 void insertLast_21104063(list &l, Node* P) {
42     if (l.tail == nullptr) {
43         l.head = P;
44         l.tail = P;
45     } else {
46         P->prev = l.tail;
47         l.tail->next = P;
48         l.tail = P;
49     }
50 }
51
52 void insertSorted_21104063(list &l, Node* P) {
53     if (l.head == nullptr || l.head->data >= P->data) {
54         P->next = l.head;
55         if (l.head != nullptr) {
56             l.head->prev = P;
57         }
58         l.head = P;
59     } else if (l.tail == nullptr) {
60         l.tail = P;
61     } else {
62         Node* current = l.head;
63         while (current->next != nullptr && current->next->data < P->data) {
64             current = current->next;
65         }
66         P->next = current->next;
67         if (current->next != nullptr) {
68             current->next->prev = P;
69         }
70         current->next = P;
71         P->prev = current;
72     }
73 }
74
75 void deleteFirst_21104063(list &l) {
76     if (l.head == nullptr) {
77         cout << "List kosong, tidak ada elemen yang bisa dihapus." << endl;
78     } else if (l.head == l.tail) {
79         delete l.head;
80         l.head = nullptr;
81         l.tail = nullptr;
82     } else {
83         Node* temp = l.head;
84         l.head = l.head->next;
85         l.head->prev = nullptr;
86         delete temp;
87     }
88 }
89
90 void deleteLast_21104063(list &l) {
91     if (l.tail == nullptr) {
92         cout << "List kosong, tidak ada elemen yang bisa dihapus." << endl;
93     } else if (l.head == l.tail) {
94         delete l.head;
95         l.head = nullptr;
96         l.tail = nullptr;
97     } else {
98         Node* current = l.head;
99         while (current->next != nullptr) {
100             current = current->next;
101         }
102         current->next = nullptr;
103         delete current;
104         l.tail = current;
105     }
106 }
107
108 void printListFromHead_21104063(list l) {
109     if (l.head == nullptr) {
110         cout << "List kosong." << endl;
111     } else {
112         Node* temp = l.head;
113         while (temp != nullptr) {
114             cout << temp->data;
115             if (temp->next != nullptr) {
116                 cout << " <-> ";
117             }
118             temp = temp->next;
119         }
120         cout << endl;
121     }
122 }
123
124 void inputElements_21104063(list &l) {
125     for (int i = 1; i <= 4; i++) {
126         int value;
127         cout << "Masukkan elemen ke-" << i << " : ";
128         cin >> value;
129         Node* newNode = alokasi_21104063(value);
130         insertSorted_21104063(l, newNode);
131     }
132 }
133
134 int main() {
135     list l;
136     createlist_21104063(l);
137     inputElements_21104063(l);
138     cout << "\nList setelah elemen dimasukkan secara terurut:" << endl;
139     printListFromHead_21104063(l);
140
141     int newValue;
142     cout << "\nMasukkan elemen tambahan yang ingin dimasukkan:" << endl;
143     cin >> newValue;
144     Node* newNode = alokasi_21104063(newValue);
145     insertSorted_21104063(l, newNode);
146     cout << "\nList setelah elemen baru dimasukkan:" << endl;
147     printListFromHead_21104063(l);
148     return 0;
149 }
```

Output :

```
ub.exe --interpreter=msi
Masukkan elemen ke-1: 89
Masukkan elemen ke-2: 65
Masukkan elemen ke-3: 34
Masukkan elemen ke-4: 77

List setelah elemen dimasukkan secara terurut:
34 65 77 89

Masukkan elemen tambahan yang ingin dimasukkan secara terurut: 76

List setelah elemen baru dimasukkan:
34 65 76 77 89
PS D:\= Collage\Semester 7\Praktikum Struktur Data\STD_Hamidatun_Nisa_21104063>
```