

LAPORAN PRAKTIKUM
PERTEMUAN 5
SINGLE LINKED LIST BAGIAN KEDUA



Nama :

Yehuda Melvin Sugiarto (2311104055)

Dosen :

Wahyu Andi Saputra

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

I. 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

II. TOOL

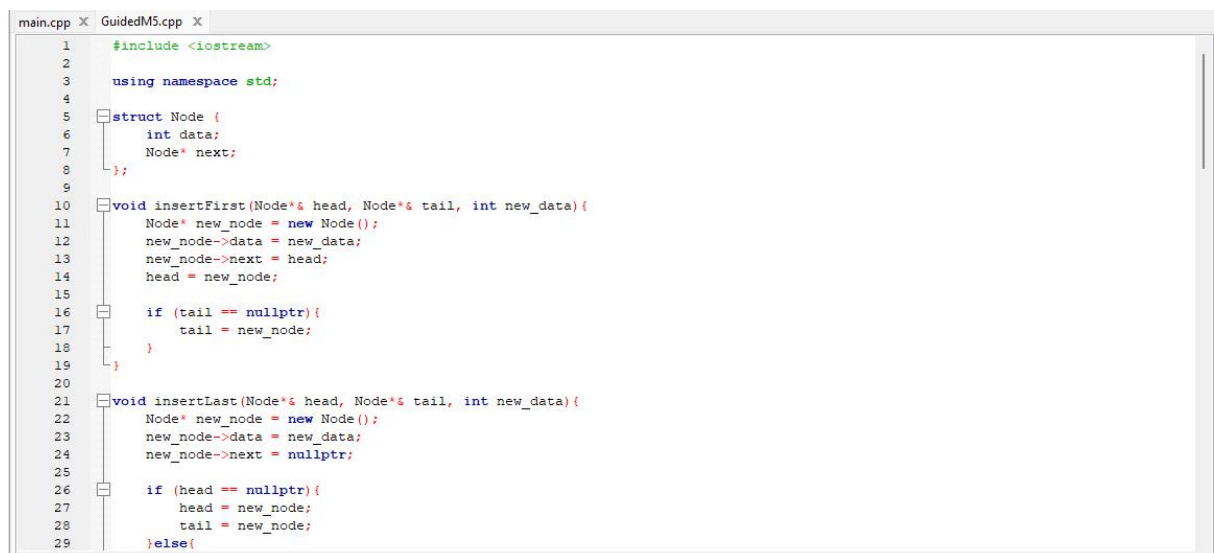
- C++
- Code::Block
- Laptop

III. DASAR TEORI

A. Searching

Searching adalah operasi dasar dalam daftar yang bertujuan mencari node tertentu. Proses ini dilakukan dengan mengunjungi setiap node satu per satu hingga node yang dicari ditemukan. Melalui operasi pencarian, berbagai operasi lain seperti penyisipan setelah node, penghapusan setelah node, dan pembaruan menjadi lebih mudah dilakukan.

IV. GUIDE



```
main.cpp x GuidedM5.cpp x
1      #include <iostream>
2
3      using namespace std;
4
5      struct Node {
6          int data;
7          Node* next;
8      };
9
10     void insertFirst(Node*& head, Node*& tail, int new_data){
11         Node* new_node = new Node();
12         new_node->data = new_data;
13         new_node->next = head;
14         head = new_node;
15
16         if (tail == nullptr){
17             tail = new_node;
18         }
19     }
20
21     void insertLast(Node*& head, Node*& tail, int new_data){
22         Node* new_node = new Node();
23         new_node->data = new_data;
24         new_node->next = nullptr;
25
26         if (head == nullptr){
27             head = new_node;
28             tail = new_node;
29         }
30     }
```

```

main.cpp x GuidedM5.cpp x
30     tail->next = new_node;
31     tail = new_node;
32 }
33
34
35 int findElement(Node* head, int x){
36     Node* current = head;
37     int index = 0;
38
39     while (current != nullptr){
40         if (current->data == x){
41             return index;
42         }
43         current = current->next;
44         index++;
45     }
46     return -1;
47 }
48
49 void display(Node* node){
50     while (node != nullptr){
51         cout << node->data << " ";
52         node = node->next;
53     }
54     cout << endl;
55 }
56
57 void deleteElement(Node*& head, int x){
58     if (head == nullptr){
59         cout << "Linked List Kosong" << endl;
60         return;
61     }
62
63     if (head->data == x){
64         Node* temp = head;
65         head = head->next;
66         delete temp;
67         return;
68     }
69
70     Node* current = head;
71     while (current->next != nullptr){
72         if (current->next->data == x){
73             Node* temp = current->next;
74             current->next = current->next->next;
75             delete temp;
76             return;
77         }
78         current = current->next;
79     }
80 }
81
82 int main() {
83     Node* head = nullptr;
84     Node* tail = nullptr;
85
86     insertFirst(head, tail, 3);
87     insertFirst(head, tail, 5);
88
89     insertFirst(head, tail, 7);
90
91     insertLast(head, tail, 11);
92     insertLast(head, tail, 14);
93     insertLast(head, tail, 18);
94
95     cout << "Elemen dalam Linked List : ";
96     display(head);
97
98     int x;
99     cout << "Masukan elemen yang dicari : ";
100    cin >> x;
101
102    int result = findElement(head, x);
103
104    if (result == -1)
105        cout << "Elemen tidak ditemukan dalam Linked List" << endl;
106    else
107        cout << "Elemen ditemukan pada index " << result << endl;
108
109    cout << "Masukan elemen yang ingin dihapus : ";
110    cin >> x;
111    deleteElement(head, x);
112
113    cout << "Elemen dalam Linked List setelah penghapusan : ";
114    display(head);
115
116    return 0;

```

Dikodingan berikut, terdapat beberapa fungsi didalamnya, yang pertama kita membuat struktur dari Node itu sendiri terlebih dahulu, lalu kita buat beberapa fungsi insert. Yang pertama kita buat fungsi insertFirst() dimana digunakan untuk menambahkan data di awal list (sebelum head), lalu kita juga menambahkan insertLast() yang dimana berfungsi untuk menambahkan data/node di akhir list (setelah tail). selanjutnya kita juga menambahkan fungsi findElement() yang digunakan untuk mencari isi dari data/list/node yang di inginkan, di fungsi ini kita menggunakan perulangan (looping) untuk mencari data tersebut. Lalu kita juga

menambahkan fungsi display untuk menampilkan data yang kita miliki. Lalu, kita juga punya fungsi deleteElement() yang digunakan untuk menghapus elemen yang diinginkan. Yang terakhir, difungsi main digunakan untuk mengimplementasikan seluruh fungsi yang ada, yang dimana kita sudah menambahkan list diawal, lalu memunculkan list yang dimiliki, lalu mencari list yang diinginkan, lalu fungsi penghapusan dijalankan di terakhir.

V. UNGUIDED

- Membuat ADT, yang dimana output akan memunculkan isi elemen, mencari elemen dengan address yang ditentukan, dan menambahkan seluruh isi elemen yang ada.

file .h

```
main.cpp x GuidedM5.cpp x singlelist.h x singlelist.cpp x main.cpp x
2  #define SINGLELIST_H
3
4  #include <iostream>
5  using namespace std;
6
7  typedef int infotype;
8  typedef struct ElmList *address;
9
10 struct ElmList {
11     infotype info;
12     address next;
13 };
14
15 struct List {
16     address First;
17 };
18
19 address findElm(const List &L, infotype x);
20 int sumInfo(const List &L);
21
22 void createList(List &L);
23 address alokasi(infotype x);
24 void dealokasi(address &P);
25 void printInfo(const List &L);
26 void insertFirst(List &L, address P);
27
28 #endif
29
```

file .cpp

```
main.cpp x GuidedM5.cpp x singlelist.h x singlelist.cpp x main.cpp x
1  #include "singlelist.h"
2
3  void createList(List &L) {
4      L.First = NULL;
5  }
6
7  address alokasi(infotype x) {
8      address P = new ElmList;
9      if (P != NULL) {
10         P->info = x;
11         P->next = NULL;
12     }
13     return P;
14 }
15
16 void dealokasi(address &P) {
17     delete P;
18     P = NULL;
19 }
20
21 void insertFirst(List &L, address P) {
22     if (L.First == NULL) {
23         L.First = P;
24     } else {
25         P->next = L.First;
26         L.First = P;
27     }
28 }
29
```

```

main.cpp x GuidedM5.cpp x singlelist.h x singlelist.cpp x main.cpp x
26     L.First = P;
27 }
28
29
30 void printInfo(const List &L) {
31     address P = L.First;
32     while (P != NULL) {
33         cout << P->info << " ";
34         P = P->next;
35     }
36     cout << endl;
37 }
38
39 address findElm(const List &L, infotype x) {
40     address P = L.First;
41     while (P != nullptr) {
42         if (P->info == x) {
43             return P;
44         }
45         P = P->next;
46     }
47     return nullptr;
48 }
49

```

```

50 int sumInfo(const List &L) {
51     address P = L.First;
52     int sum = 0;
53     while (P != nullptr) {
54         sum += P->info;
55         P = P->next;
56     }
57     return sum;
58 }
59

```

file main

```

main.cpp x GuidedM5.cpp x singlelist.h x singlelist.cpp x main.cpp x
1  #include <iostream>
2  #include "singlelist.h"
3
4  using namespace std;
5
6  int main() {
7      List L;
8      address P1, P2, P3, P4, P5;
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

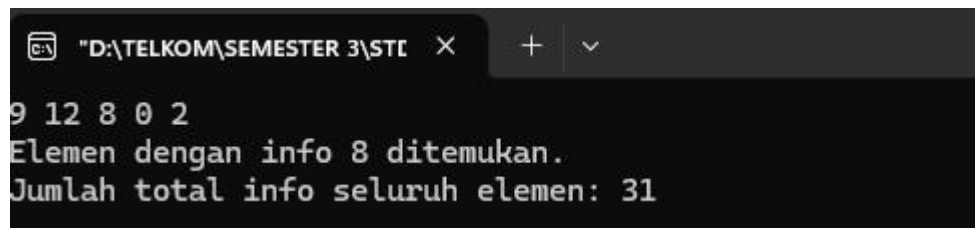
```

```

26     printInfo(L);
27
28     address found = findElm(L, 8);
29     if (found != nullptr) {
30         cout << "Elemen dengan info 8 ditemukan." << endl;
31     } else {
32         cout << "Elemen dengan info 8 tidak ditemukan." << endl;
33     }
34
35     int total = sumInfo(L);
36     cout << "Jumlah total info seluruh elemen: " << total << endl;
37
38     return 0;
39 }
40
41

```

Output



```
"D:\TELKOM\SEMESTER 3\STC" X + v
9 12 8 0 2
Elemen dengan info 8 ditemukan.
Jumlah total info seluruh elemen: 31
```

VI. KESIMPULAN

Praktikum kali ini dilakukan untuk mengetahui apa itu searching dan implementasinya menggunakan insert yang memiliki 2 tipe, insert first dan insert last. Dan implementasi lainnya seperti delete dan find element. Kita juga belajar bagaimana penggunaan search untuk menjumlahkan seluruh isi elemen yang dimiliki ADT yang kita punya.