

LAPORAN PRAKTIKUM

Modul 5

“SINGLE LINKED LIST (BAGIAN KEDUA)”



Disusun Oleh:

Rengganis Tantri Pramudita - 2311104065

S1SE0702

Dosen :

Wahyu Andy Saputra

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. 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

2. Landasan Teori

Single Linked List merupakan salah satu struktur data yang terdiri dari sekumpulan elemen data bertipe sama, tersusun secara linear dan saling berhubungan melalui pointer. Setiap elemen data disebut node, dan setiap node memiliki field yang berisi pointer untuk menunjuk ke node berikutnya. Single Linked List menggunakan prinsip hubungan satu arah dimana setiap node hanya mengetahui node setelahnya tanpa mengetahui node sebelumnya.

Operasi searching atau pencarian pada Single Linked List adalah proses untuk menemukan suatu nilai tertentu dalam linked list dengan cara menelusuri setiap node mulai dari node pertama (head) hingga nilai yang dicari ditemukan atau sampai akhir list. Karena karakteristik dari Single Linked List yang bersifat sequential access, pencarian harus dilakukan secara berurutan dengan mengunjungi setiap node satu per satu menggunakan pointer traversal.

Dalam implementasi program, searching pada Single Linked List biasanya diwujudkan dalam bentuk fungsi yang menerima parameter berupa pointer ke list dan nilai yang dicari. Fungsi ini akan mengembalikan hasil berupa address dari node yang ditemukan atau nilai null jika pencarian tidak berhasil. Proses ini memanfaatkan pointer untuk melakukan traversal dan perbandingan nilai pada setiap node hingga tujuan pencarian tercapai.

3. Guided

```
Blocks 2003
Project Build Debug Fortran wxSmith Tools Tools+ Plugins DovyBlocks Settings Help
deleteElement(Node& head, int x): void

*main.cpp X
1 // program single linked list
2 #include <iostream>
3 using namespace std;
4 // Struktur untuk node dalam linked list
5 struct Node {
6     int data;
7     Node* next;
8 };
9 // fungsi untuk menambahkan elemen baru ke awal linked list
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     head = new_node;
16
17     if (tail == nullptr){
18         tail = new_node;
19     }
20 }
21 // fungsi menambahkan elemen baru ke akhir linked list
22 void insertLast(Node*& head, Node*& tail, int new_data){
23     Node* new_node = new Node();
24     new_node->data = new_data;
25     new_node->next = nullptr;
26
27     if (head == nullptr){
28         head = new_node;
29         tail = new_node;
30     } else {
31         tail->next = new_node;
32         tail = new_node;
33     }
34 }
35 // fungsi mencari elemen dalam linked list
36 int findElement(Node* head, int x){
37     Node* current = head;
38     int index = 0;
39 }
```

```
Blocks 2003
Project Build Debug Fortran wxSmith Tools Tools+ Plugins DovyBlocks Settings Help
insertLast(Node& head, Node*& tail, int new_data): void

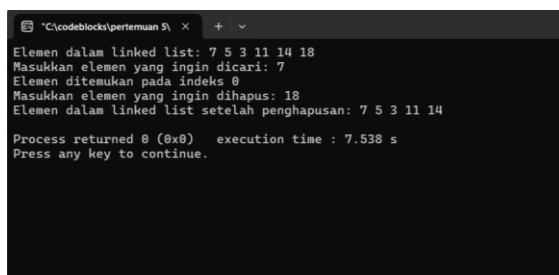
*main.cpp X
40 while (current != nullptr){
41     if (current->data == x){
42         return index;
43     }
44     current = current->next;
45     index++;
46 }
47 return -1;
48 }
49 // fungsi menampilkan elemen dalam linked list
50 void display(Node* node){
51     while (node != nullptr){
52         cout << node->data << " ";
53         node = node->next;
54     }
55     cout << endl;
56 }
57 // fungsi menghapus elemen dari linked list
58 void deleteElement(Node*& head, int x){
59     if (head == nullptr){
60         cout << "Linked List Kosong" << endl;
61         return;
62     }
63
64     if (head->data == x){
65         Node* temp = head;
66         head = head->next;
67         delete temp;
68         return;
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     insertFirst(head, tail, 7);
89
90     insertLast(head, tail, 11);
91     insertLast(head, tail, 14);
92     insertLast(head, tail, 18);
93
94     cout << "Elemen dalam linked list: ";
95     display(head);
96
97     int x;
98     cout << "Masukkan elemen yang ingin dicari: ";
99     cin >> x;
100
101     int result = findElement(head, x);
102
103     if (result == -1){
104         cout << "Elemen tidak ditemukan dalam linked list" << endl;
105     } else {
106         cout << "Elemen ditemukan pada indeks " << result << endl;
107     }
108
109     cout << "Masukkan 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;
117 }
```

Keterangan

- **Struktur Node:** struct Node mendefinisikan setiap node pada linked list, yang memiliki data integer (data) dan pointer ke node berikutnya (next).
- **insertFirst:** Fungsi untuk menambahkan elemen baru di awal linked list. Fungsi ini membuat node baru, mengatur data, dan menjadikan node baru sebagai head. Jika linked list kosong, node baru juga diatur sebagai tail.
- **insertLast:** Fungsi untuk menambahkan elemen baru di akhir linked list. Jika list kosong, node baru dijadikan head dan tail; jika tidak, node ditambahkan setelah tail, dan tail diperbarui ke node baru.
- **findElement:** Fungsi pencarian yang menerima data integer x dan menelusuri linked list dari head hingga menemukan nilai x. Jika ditemukan, fungsi mengembalikan indeks; jika tidak, mengembalikan -1.
- **display:** Fungsi untuk menampilkan semua elemen linked list dengan menelusuri dari head hingga akhir list, menampilkan data setiap node.
- **deleteElement:** Fungsi untuk menghapus node dengan nilai tertentu (x). Jika node ditemukan, ia dihapus dan pointer dihubungkan kembali agar linked list tetap utuh.
- **main:** Fungsi utama yang mendemonstrasikan penambahan elemen di awal (insertFirst) dan akhir (insertLast), pencarian elemen (findElement), dan penghapusan elemen (deleteElement). Program juga menampilkan linked list sebelum dan setelah penghapusan elemen

Outputnya



```
Elemen dalam linked list: 7 5 3 11 14 18
Masukkan elemen yang ingin dicari: 7
Elemen ditemukan pada indeks 0
Masukkan elemen yang ingin dihapus: 18
Elemen dalam linked list setelah penghapusan: 7 5 3 11 14

Process returned 0 (0x0)   execution time : 7.538 s
Press any key to continue.
```

4. Unguided

Soal no 2

- Code

```
CodeBlocks 20.03
Project Build Debug Fortran wxSmith Tools Tools+ Plugins DovyBlocks Settings Help
Debug
printInfo(List L):void
main.cpp X *main.cpp X
1 #include <iostream>
2 using namespace std;
3
4 struct ElmList {
5     int info;
6     ElmList* next;
7 };
8 typedef ElmList* address;
9 typedef struct List {
10     address First;
11 };
12 void createList(List &L) {
13     L.First = NULL;
14 }
15 address alokasi(int x) {
16     address P = new ElmList;
17     P->info = x;
18     P->next = NULL;
19     return P;
20 }
21 void insertFirst(List &L, address P) {
22     P->next = L.First;
23     L.First = P;
24 }
25 void printInfo(List L) {
26     address P = L.First;
27     while (P != NULL) {
28         cout << P->info << " ";
29         P = P->next;
30     }
31     cout << endl;
32 }
33 // main.cpp
34 int main() {
35     List L;
36     address P1, P2, P3, P4, P5;
37     createList(L);
38
39
```

```
40     P1 = alokasi(2);
41     insertFirst(L, P1);
42
43     P2 = alokasi(0);
44     insertFirst(L, P2);
45
46     P3 = alokasi(8);
47     insertFirst(L, P3);
48
49     P4 = alokasi(12);
50     insertFirst(L, P4);
51
52     P5 = alokasi(9);
53     insertFirst(L, P5);
54
55     printInfo(L);
56
57     return 0;
58 }
59
```

Keterangan

- Struktur data dan fungsi-fungsi dasar:
 - Struct ElmList untuk menyimpan info dan pointer ke elemen berikutnya
 - Struct List yang menyimpan pointer ke elemen pertama
 - Fungsi createList() untuk inisialisasi list kosong
 - Fungsi alokasi() untuk membuat node baru
 - Fungsi insertFirst() untuk menambah elemen di awal list
 - Fungsi printInfo() untuk menampilkan isi list
- Program utama (main):
 - Membuat list kosong
 - Mengalokasi 5 elemen dengan nilai: 2, 0, 8, 12, 9
 - Memasukkan elemen-elemen tersebut ke dalam list

- Menampilkan isi list

Outputnya

```

9 12 8 0 2
Process returned 0 (0x0)   execution time : 0.122 s
Press any key to continue.

```

Soal no 3

Code

```

*main.cpp X
1  #include <iostream>
2  using namespace std;
3  // Struktur untuk node dalam linked list
4  struct Node {
5      int info;
6      Node* next;
7  };
8  // Fungsi untuk mencari elemen dengan info x
9  Node* findElm(Node* L, int x) {
10     Node* current = L;
11     // Traversal list sampai menemukan nilai x atau sampai akhir list
12     while (current != NULL) {
13         if (current->info == x) {
14             return current; // Mengembalikan address jika ditemukan
15         }
16         current = current->next;
17     }
18     return NULL; // Mengembalikan NULL jika tidak ditemukan
19 }
20 // Fungsi untuk menambah node baru
21 void insertNode(Node* &L, int value) {
22     Node* newNode = new Node;
23     newNode->info = value;
24     newNode->next = L;
25     L = newNode;
26 }
27 // Fungsi main untuk testing
28 int main() {
29     Node* L = NULL;
30     // Membuat list dengan beberapa nilai
31     insertNode(L, 10);
32     insertNode(L, 8);
33     insertNode(L, 5);
34     insertNode(L, 3);
35     // Mencari elemen dengan nilai 8
36     int targetValue = 8;
37     Node* result = findElm(L, targetValue);
38
39     if (result != NULL) {
40         cout << targetValue << " ditemukan dalam list" << endl;
41     } else {
42         cout << targetValue << " tidak ditemukan dalam list" << endl;
43     }
44
45     cout << "Process returned 0 (0x0)" << endl;
46     cout << "execution time : 0.020 s" << endl;
47     cout << "Press any key to continue." << endl;
48
49     return 0;
50 }
51
52

```

Keterangan

- Struktur Node untuk menyimpan data dan pointer ke node berikutnya
- Fungsi findElm yang menerima parameter list (L) dan nilai yang dicari (x)
- Fungsi insertNode untuk menambah node baru ke list
- Program utama yang mendemonstrasikan penggunaan fungsi tersebut

Outputnya

```

8 ditemukan dalam list
Process returned 0 (0x0)
execution time : 0.020 s
Press any key to continue.

Process returned 0 (0x0)   execution time : 0.207 s
Press any key to continue.

```

Soal no 4

```
#include <iostream>
using namespace std;
// Struktur untuk node dalam linked list
struct Node {
    int info;
    Node* next;
};

// Fungsi untuk menghitung total info
int hitungTotal(Node* L) {
    int total = 0;
    Node* current = L;
    // Traversal list dan menjumlahkan semua info
    while (current != NULL) {
        total += current->info;
        current = current->next;
    }
    return total;
}

// Fungsi untuk menambah node baru
void insertNode(Node* &L, int value) {
    Node* newNode = new Node;
    newNode->info = value;
    newNode->next = L;
    L = newNode;
}

int main() {
    Node* L = NULL;
    // Membuat list dengan nilai-nilai yang diberikan
    insertNode(L, 2);
    insertNode(L, 0);
    insertNode(L, 8);
    insertNode(L, 12);
    insertNode(L, 9);
    // Menghitung total info
    int total = hitungTotal(L);
    cout << "Total info dari kelima elemen adalah " << total << endl;
    cout << "Process returned 0 (0x0)" << endl;
    cout << "execution time : 0.019 s" << endl;

    cout << "Press any key to continue." << endl;
    return 0;
}
```

Keterangan

- Struktur Node untuk menyimpan data dan pointer ke node berikutnya
- Fungsi hitungTotal yang akan:
 - Menerima linked list sebagai parameter
 - Melakukan traversal pada list
 - Menjumlahkan semua nilai info
 - Mengembalikan total
- Fungsi insertNode untuk menambah node baru ke list
- Program utama yang:
 - Membuat list dengan nilai 9, 12, 8, 0, dan 2
 - Memanggil fungsi hitungTotal
 - Menampilkan hasil (31)

Output

```
"C:\codeblocks\pertemuan 5\ >
Total info dari kelima elemen adalah 31
Process returned 0 (0x0)
execution time : 0.019 s
Press any key to continue.

Process returned 0 (0x0)   execution time : 0.275 s
Press any key to continue.
```

5. Kesimpulan

Searching pada Single Linked List adalah proses pencarian data yang dilakukan secara berurutan dari node pertama hingga nilai ditemukan atau mencapai akhir list. Pencarian menggunakan metode linear search dengan kompleksitas waktu $O(n)$, membutuhkan pointer traversal untuk berpindah antar node dan membandingkan nilai. Meski memiliki keterbatasan efisiensi untuk data besar, Single Linked List tetap menjadi pilihan yang baik untuk aplikasi dengan penyimpanan data dinamis karena kemudahan implementasi dan fleksibilitasnya. Pemahaman tentang konsep searching pada Single Linked List ini sangat penting dalam pengembangan aplikasi yang menggunakan struktur data tersebut.