

LAPORAN PRAKTIKUM
Modul 5
“Single Linked List (Bagian Kedua)”



Disusun Oleh:

Ahmad Al - Farizi - 2311104054

Kelas :

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd, M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. 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.

2. Landasan Teori

2.1. Searching

Searching adalah salah satu operasi dasar yang sangat penting dalam pengelolaan data, terutama ketika bekerja dengan struktur data seperti list. Proses pencarian ini melibatkan upaya untuk menemukan elemen atau node tertentu dengan cara mengunjungi setiap node satu per satu, yang dikenal sebagai pencarian linier (linear search). Dalam proses ini, dimulai dari node pertama dalam list, membandingkan nilai node saat ini dengan nilai yang dicari. Jika tidak cocok, akan berpindah ke node berikutnya hingga node yang dicari ditemukan atau akhir list tercapai. Pentingnya operasi searching terletak pada kemampuannya untuk mempermudah berbagai operasi lain, seperti insert after, delete after, dan update.

Setelah menemukan node tertentu, elemen baru dapat ditambahkan setelahnya dengan mengubah pointer dari node tersebut. Begitu juga, dapat menghapus node berikutnya atau memperbarui nilai dari suatu node setelah menemukannya melalui proses pencarian. Meskipun pencarian linier sederhana dan mudah dipahami, ia memiliki kelemahan dalam hal efisiensi, terutama ketika berhadapan dengan list yang sangat besar. Untuk meningkatkan efisiensi pencarian, beberapa teknik dapat diterapkan, seperti menggunakan struktur data lain seperti hash table atau binary search tree, serta mengurutkan list sebelum melakukan pencarian untuk memungkinkan penggunaan algoritma pencarian yang lebih cepat seperti binary search.

Dengan demikian, pemahaman mendalam tentang cara kerja searching dan implikasinya terhadap operasi lain sangatlah krusial dalam dunia pemrograman dan pengelolaan data.

3. Guided

3.1. Modul 5

Program ini mengimplementasikan linked list sederhana dengan fungsi untuk menambah, mencari, menampilkan, dan menghapus elemen. Struktur `Node` menyimpan data dan pointer ke elemen berikutnya, dengan `head` menunjuk ke elemen pertama dan `tail` ke elemen terakhir. Fungsi `insertFirst` menambah elemen di awal list, sedangkan `insertLast` menambah di akhir. Fungsi `findElement` mencari elemen berdasarkan nilai, dan `deleteElement` menghapus elemen tertentu dari list. Program juga menampilkan semua elemen dengan fungsi `display`. Pada `main()`, beberapa elemen ditambahkan, lalu program meminta input pengguna untuk mencari dan menghapus elemen, serta menampilkan hasilnya.

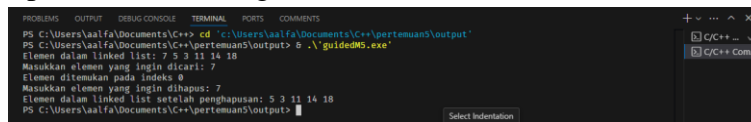
Kode Program :

```
1  #include <iostream>
2  using namespace std;
3
4  // struktur untuk node dalam linked list
5  struct Node {
6      int data;
7      Node* next;
8  };
9
10 // fungsi untuk menambahkan elemen baru ke awal linked list
11 void insertFirst(Node*& head, Node*& tail, int new_data){
12     Node* new_node = new Node();
13     new_node->data = new_data;
14     new_node->next = head;
15     head = new_node;
16
17     if (tail == nullptr) {
18         tail = new_node;
19     }
20 }
21
22 // fungsi untuk menambahkan elemen baru ke akhir linked list
23 void insertLast(Node*& head, Node*& tail, int new_data){
24     Node* new_node = new Node ();
25     new_node->data = new_data;
26     new_node->next = nullptr;
27
28     if (head == nullptr){
29         head = new_node;
30         tail = new_node;
31     } else {
32         tail->next = new_node;
33         tail = new_node;
34     }
35 }
36
```

```
1 // fungsi untuk mencari elemen dalam linked list
2 int findElement(Node* head, int x){
3     Node* current = head;
4     int index = 0;
5
6     while(current != nullptr){
7         if (current->data == x){
8             return index;
9         }
10        current = current->next;
11        index++;
12    }
13    return -1;
14 }
15
16 // fungsi untuk menampilkan elemen dalam linked list
17 void display(Node* node){
18     while (node != nullptr){
19         cout << node->data << " ";
20         node = node->next;
21     }
22     cout << endl;
23 }
24
25 // fungsi untuk menghapus elemen dari linked list
26 void deleteElement(Node* head, int x){
27     if (head == nullptr){
28         cout << "Linked List kosong" << endl;
29         return;
30     }
31
32     if (head->data == x){
33         Node* temp = head;
34         head = head->next;
35         delete temp;
36         return;
37     }
38
39     Node* current = head;
40     while(current->next != nullptr){
41         if(current->next->data == x){
42             Node* temp = current->next;
43             current->next = current->next->next;
44             delete temp;
45             return;
46         }
47         current = current->next;
48     }
49 }
50
```

```
1  int main()
2  {
3      Node* head = nullptr;
4      Node* tail = nullptr;
5
6      insertFirst(head, tail, 3);
7      insertFirst(head, tail, 5);
8      insertFirst(head, tail, 7);
9
10     insertLast(head, tail, 11);
11     insertLast(head, tail, 14);
12     insertLast(head, tail, 18);
13
14     cout << "Elemen dalam linked list: ";
15     display(head);
16
17     int x;
18     cout << "Masukkan elemen yang ingin dicari: ";
19     cin >> x;
20
21     int result = findElement(head, x);
22
23     if (result == -1)
24         cout << "Elemen tidak ditemukan dalam linked list" << endl;
25     else
26         cout << "Elemen ditemukan pada indeks " << result << endl;
27
28     cout << "Masukkan elemen yang ingin dihapus: ";
29     cin >> x;
30     deleteElement(head, x);
31
32     cout << "Elemen dalam linked list setelah penghapusan: ";
33     display(head);
34
35     return 0;
36 }
```

Output dari Kode Program :



```
PS C:\Users\aaifa\Documents\C++> cd "c:\Users\aaifa\Documents\C++\pertemuan5\output"
PS C:\Users\aaifa\Documents\C++\pertemuan5\output> g .\guided05.exe
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: 7
Elemen dalam linked list setelah penghapusan: 5 3 11 14 18
PS C:\Users\aaifa\Documents\C++\pertemuan5\output>
```

4. Unguided

4.1. Kode Program :

```
singlelist.h X main.cpp X singlelist.cpp X
1 #ifndef SINGLELIST_H
2 #define SINGLELIST_H
3
4 // Deklarasi tipe data
5 typedef int infotype;
6 typedef struct ElmtList *address;
7
8 struct ElmtList {
9     infotype info;
10    address next;
11 }
12
13 struct List {
14     address First;
15 }
16
17 // Deklarasi prosedur dan fungsi
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
24 #endif
25
```

```
singlelist.cpp X main.cpp X
1 #include <iostream>
2 #include "singlelist.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 ElmtList;
12     P->info = x;
13     P->next = NULL;
14     return P;
15 }
16
17 void dealokasi(address &P) {
18     delete P;
19     P = NULL;
20 }
21
22 void printInfo(List L) {
23     address P = L.First;
24     while (P != NULL) {
25         cout << P->info << " ";
26         P = P->next;
27     }
28     cout << endl;
29 }
30
31 void insertFirst(List &L, address P) {
32     P->next = L.First;
33     L.First = P;
34 }
35
```

```
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(3);
19    insertFirst(L, P3);
20
21    P4 = alokasi(10);
22    insertFirst(L, P4);
23
24    P5 = alokasi(9);
25    insertFirst(L, P5);
26
27    printInfo(L);
28
29    return 0;
30 }
31
```

Output dari Kode Program :

```

C:\Users\user\Documents\Coding\Unguided Modul 3\Unguided Modul 3\Unguided Modul 3.exe
0 12 0 0
Process returned 0 (0x0)   execution time : 0.208 s
Press any key to continue.

```

4.2. Kode Program :

```

1  #ifndef SINGLELIST_H
2  #define SINGLELIST_H
3
4  // Deklarasi tipe data
5  typedef int infotype;
6  typedef struct ElmtList *address;
7
8  struct ElmtList {
9      infotype info;
10     address next;
11 };
12
13 struct List {
14     address First;
15 };
16
17 // Deklarasi prosedur dan fungsi
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
25 #endif
26

```

```

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 ElmtList;
12     P->info = x;
13     P->next = NULL;
14     return P;
15 }
16
17 void dealokasi(address &P) {
18     delete P;
19     P = NULL;
20 }
21
22 void printInfo(List L) {
23     address P = L.First;
24     while (P != NULL) {
25         cout << P->info << " ";
26         P = P->next;
27     }
28     cout << endl;
29 }
30
31 void insertFirst(List &L, address P) {
32     P->next = L.First;
33     L.First = P;
34 }
35
36 address findElm(List L, infotype x) {
37     address P = L.First;
38     while (P != NULL) {
39         if (P->info == x) {
40             return P;
41         }
42         P = P->next;
43     }
44     return NULL;
45 }
46

```

```

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
27     printInfo(L);
28
29     infotype x = 5;
30     address found = findElm(L, x);
31     if (found != NULL) {
32         cout << x << " ditemukan dalam list" << endl;
33     } else {
34         cout << x << " tidak ditemukan dalam list" << endl;
35     }
36     return 0;
37 }
38

```

Ouput dari Kode Program :

```

C:\Users\aulia\Documents\C++\Uniguided Modul 9\Uniguided Modul 9\bin\Debug\Uniguided Modul 9.exe
0 12 0 1
5 ditemukan dalam list
Process returned 0 (0x0)   execution time : 0.238 s
Press any key to continue.

```

4.3. Kode Program :

```
singlelist.h X singlelist.cpp X main.cpp X
1  #ifndef SINGLELIST_H
2  #define SINGLELIST_H
3
4  // Deklarasi tipe data
5  typedef int infotype;
6  typedef struct ElmtList *address;
7
8  struct ElmtList {
9      infotype info;
10     address next;
11 }
12
13 struct List {
14     address First;
15 }
16
17 // Deklarasi prosedur dan fungsi
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 int sumInfo(List L);
24
25 #endif
```

```
singlelist.h X singlelist.cpp X main.cpp X
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 ElmtList;
12     P->info = x;
13     P->next = NULL;
14     return P;
15 }
16
17 void dealokasi(address &P) {
18     delete P;
19     P = NULL;
20 }
21
22 void printInfo(List L) {
23     address P = L.First;
24     while (P != NULL) {
25         cout << P->info << " ";
26         P = P->next;
27     }
28     cout << endl;
29 }
30
31 void insertFirst(List &L, address P) {
32     P->next = L.First;
33     L.First = P;
34 }
35
36 int sumInfo(List L) { // Implementasi fungsi sumInfo
37     int sum = 0;
38     address P = L.First;
39     while (P != NULL) {
40         sum += P->info;
41         P = P->next;
42     }
43     return sum;
44 }
```

```
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(3);
16     insertFirst(L, P2);
17
18     P3 = alokasi(4);
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     int total = sumInfo(L); // Implementasi fungsi sumInfo
30     cout << "Total info dari kelima elemen adalah " << total << endl;
31     return 0;
32 }
33
```

Output dari Kode Program :

```

C:\Users\adfa\Documents\C++\Uniguided Modul 9\Uniguided Modul 9\Uniguided Modul 9.exe
0 12 0 0
Total info dari kelima elemen adalah 31
Process returned 0 (0x0)   execution time : 0.211 s
Press any key to continue.

```


5. Kesimpulan

Searching merupakan operasi dasar yang esensial dalam pengelolaan data, khususnya pada struktur data seperti list. Proses pencarian yang dilakukan dengan mengunjungi setiap node secara linier memungkinkan pengguna untuk menemukan elemen tertentu, yang kemudian mempermudah pelaksanaan berbagai operasi lain seperti insert after, delete after, dan update. Meskipun pencarian linier sederhana, efisiensinya dapat menurun pada list yang besar. Oleh karena itu, penerapan teknik-teknik seperti penggunaan struktur data alternatif dan pengurutan list menjadi penting untuk meningkatkan kecepatan pencarian. Dengan pemahaman yang baik tentang proses searching dan implikasinya, pengguna dapat mengelola data dengan lebih efektif dan efisien dalam konteks pemrograman.

