

GUIDED LAPORAN PRAKTIKUM STUKTUR DATA MODUL 4

1. Guided

3.1. Linked List

Kode ini memiliki beberapa fungsi untuk mengelola linked list, seperti menambahkan node, menghapus node, menampilkan isi linked list, dan menghapus seluruh node. Kode ini juga memiliki sebuah struct mahasiswa untuk merepresentasikan data mahasiswa. Dalam fungsi main(), kode ini melakukan beberapa operasi seperti menambahkan node, menampilkan isi linked list, menghapus node, dan menghapus seluruh node. Dengan demikian, kode ini dapat digunakan untuk mengelola data mahasiswa dalam sebuah linked list.

Kode program :

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  // Deklarasi Struct untuk mahasiswa
6  struct mahasiswa {
7      char nama[30];
8      char nim[10];
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 // Pengecekan apakah list kosong
27 bool isEmpty() {
28     return head == nullptr;
29 }
```

```

1 // Tambah Depan
2 void insertDepan(const mahasiswa &data) {
3     Node *baru = new Node;
4     baru->data = data;
5     baru->next = nullptr;
6     if (isEmpty()) {
7         head = tail = baru;
8     } else {
9         baru->next = head;
10        head = baru;
11    }
12 }
13
14 // Tambah Belakang
15 void insertBelakang(const mahasiswa &data) {
16     Node *baru = new Node;
17     baru->data = data;
18     baru->next = nullptr;
19     if (isEmpty()) {
20         head = tail = baru;
21     } else {
22         tail->next = baru;
23         tail = baru;
24     }
25 }
26
27 // Hitung Jumlah List
28 int hitungList() {
29     Node *current = head;
30     int jumlah = 0;
31     while (current != nullptr) {
32         jumlah++;
33         current = current->next;
34     }
35     return jumlah;
36 }
37
38 // Hapus Depan
39 void hapusDepan() {
40     if (!isEmpty()) {
41         Node *hapus = head;
42         head = head->next;
43         delete hapus;
44         if (head == nullptr) {
45             tail = nullptr; // Jika list menjadi kosong
46         }
47     } else {
48         cout << "List kosong!" << endl;
49     }
50 }
51
52 // Hapus Belakang
53 void hapusBelakang() {
54     if (!isEmpty()) {
55         if (head == tail) {
56             delete head;
57             head = tail = nullptr; // List menjadi kosong
58         } else {
59             Node *bantu = head;
60             while (bantu->next != tail) {
61                 bantu = bantu->next;
62             }
63             delete tail;
64             tail = bantu;
65             tail->next = nullptr;
66         }
67     } else {
68         cout << "List kosong!" << endl;
69     }
70 }

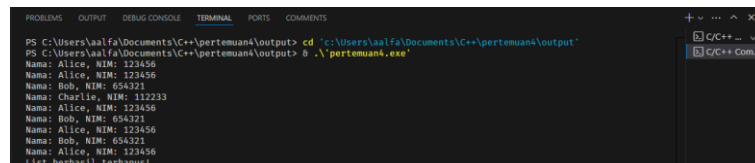
```

```

1 // Tampilkan List
2 void tampil() {
3     Node *current = head;
4     if (!isEmpty()) {
5         while (current != nullptr) {
6             cout << "Nama: " << current->data.nama << ", NIM: " << current->data.nim << endl;
7             current = current->next;
8         }
9     } else {
10        cout << "List masih kosong!" << endl;
11    }
12 }
13
14 // Hapus List
15 void clearList() {
16     Node *current = head;
17     while (current != nullptr) {
18         Node *hapus = current;
19         current = current->next;
20         delete hapus;
21     }
22     head = tail = nullptr;
23     cout << "List berhasil terhapus!" << endl;
24 }
25
26 // Main function
27 int main() {
28     init();
29
30     // Contoh data mahasiswa
31     mahasiswa m1 = {"Alice", "123456"};
32     mahasiswa m2 = {"Bob", "654321"};
33     mahasiswa m3 = {"Charlie", "112233"};
34
35     // Menambahkan mahasiswa ke dalam list
36     insertDepan(m1);
37     tampil();
38     insertBelakang(m2);
39     tampil();
40     insertDepan(m3);
41     tampil();
42
43     // Menghapus elemen dari list
44     hapusDepan();
45     tampil();
46     hapusBelakang();
47     tampil();
48
49     // Menghapus seluruh list
50     clearList();
51
52     return 0;
53 }

```

Output program :



```

PS C:\Users\saalifa\Documents\C++\pertemuan4\output> cd 'C:\Users\saalifa\Documents\C++\pertemuan4\output'
PS C:\Users\saalifa\Documents\C++\pertemuan4\output> g++ .\pertemuan4.exe
Nama: Alice, NIM: 123456
Nama: Alice, NIM: 123456
Nama: Bob, NIM: 654321
Nama: Charlie, NIM: 112233
Nama: Alice, NIM: 123456
Nama: Bob, NIM: 654321
Nama: Alice, NIM: 123456
Nama: Bob, NIM: 654321
Nama: Alice, NIM: 123456
List berhasil terhapus!

```

3.2. Single Linked List

Kode ini memiliki beberapa fungsi untuk mengelola linked list, seperti menambahkan node, menampilkan isi linked list, menghitung jumlah elemen, dan menghapus semua elemen dalam list. Kode ini juga memiliki fungsi untuk mengalokasikan dan dealokasikan memori untuk node baru. Dalam fungsi main, kode ini melakukan beberapa operasi untuk mengelola data dalam sebuah linked list.

Kode Program :

```
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 isListEmpty(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 (isListEmpty(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 (isListEmpty(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 }
```

```

1 // Menghapus semua elemen dalam list dan dealokasi memori
2 void clearList(Node* &head) {
3     while (head != nullptr) {
4         Node* temp = head; // Simpan pointer ke node saat ini
5         head = head->next; // Pindahkan ke node berikutnya
6         dealokasi(temp); // Dealokasi node
7     }
8 }
9
10 int main() {
11     Node* head = nullptr; // Membuat list kosong
12
13     // Menambahkan elemen ke dalam list
14     insertFirst(head, 10); // Menambahkan elemen 10 di awal list
15     insertLast(head, 20); // Menambahkan elemen 20 di akhir list
16     insertLast(head, 30); // Menambahkan elemen 30 di akhir list
17
18     // Menampilkan isi list
19     cout << "Isi List: ";
20     printList(head);
21
22     // Menampilkan jumlah elemen
23     cout << "Jumlah elemen: " << countElements(head) << endl;
24
25     // Menghapus semua elemen dalam list
26     clearList(head);
27
28     // Menampilkan isi list setelah penghapusan
29     cout << "Isi List setelah penghapusan: ";
30     printList(head);
31
32     return 0;
33 }

```

Output Program :

```

PS C:\Users\aaalfa\Documents\C++> cd 'C:\Users\aaalfa\Documents\C++\pertemuan4\output'
PS C:\Users\aaalfa\Documents\C++\pertemuan4\output> g++ '.\pertemuan4\juga.exe'
Isi List: 10 20 30
Jumlah elemen: 3
Isi List setelah penghapusan: List kosong!
PS C:\Users\aaalfa\Documents\C++\pertemuan4\output>

```