

Ketentuan Tugas Pendahuluan

- Untuk soal teori **JAWABAN DIKETIK DENGAN RAPI** dan untuk soal algoritma **SERTAKAN SCREENSHOOT CODINGAN DAN HASIL OUTPUT**.
- Deadline pengumpulan TP Modul 6 adalah **Senin, 21 Oktober 2024** pukul 06.00 WIB.
- **TIDAK ADA TOLERANSI KETERLAMBATAN, TERLAMBAT ATAU TIDAK MENGUMPULKAN TP ONLINE MAKA DIANGGAP TIDAK MENGERJAKAN.**
- **DILARANG PLAGIAT (PLAGIAT = E).**
- Kerjakan TP dengan jelas agar dapat dimengerti.
- Untuk setiap soal nama fungsi atau prosedur **WAJIB** menyertakan **NIM**, contoh: **insertFirst_130122xxxx**.
- File diupload di LMS menggunakan format **PDF** dengan ketentuan : **TP_MODX_NIM_KELAS.pdf**

Contoh:

```
int searchNode_130122xxxx (List L, int X);
```

CP:

- Raihan (089638482851)
- Kayyisa (085105303555)
- Abiya (082127180662)
- Rio (081210978384)

SELAMAT MENGERJAKAN^^

LAPORAN PRAKTIKUM
PERTEMUAN 6
DOUBLE LINKED LIST BAGIAN PERTAMA



Nama :

Tsaqif Hisyam Saputra (2311104024)

Dosen :

Yudha Islami Sulistya

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

A. Tujuan

- Memahami konsep modul linked list
- Mengaplikasikan konsep double linked list dengan menggunakan pointer dan dengan bahasa C++

B. Tools

Visual Studio Code dengan C++ Extension Pack

TUGAS PENDAHULUAN

1.

```
1 #include <iostream>
2
3 using namespace std;
4
5 struct Node {
6     int data;
7     Node* prev;
8     Node* next;
9 };
10
11 class DoublyLinkedList {
12 private:
13     Node* head;
14     Node* tail;
15 public:
16     DoublyLinkedList() {
17         head = NULL;
18         tail = NULL;
19     }
20
21     void insertFirst(int data) {
22         Node* newNode = new Node;
23         newNode->data = data;
24         newNode->prev = NULL;
25         newNode->next = head;
26
27         if (head != NULL) {
28             head->prev = newNode;
29         }
30         head = newNode;
31
32         if (tail == NULL) {
33             tail = newNode;
34         }
35     }
36
37     void insertLast(int data) {
38         Node* newNode = new Node;
39         newNode->data = data;
40         newNode->next = NULL;
41         newNode->prev = tail;
42
43         if (tail != NULL) {
44             tail->next = newNode;
45         }
46         tail = newNode;
47
48         if (head == NULL) {
49             head = newNode;
50         }
51     }
52
53     void printList() {
54         Node* current = head;
55         cout << "DAFTAR ANGGOTA LIST: ";
56         while (current != NULL) {
57             cout << current->data;
58             if (current->next != NULL) {
59                 cout << " <-> ";
60             }
61             current = current->next;
62         }
63         cout << endl;
64     }
65 };
66
67 int main() {
68     DoublyLinkedList dll;
69
70     dll.insertFirst(40); // Masukkan elemen pertama
71     dll.insertFirst(25); // Masukkan elemen kedua di awal
72     dll.insertLast(60); // Masukkan elemen ketiga di akhir
73
74     dll.printList();
75
76     return 0;
77 }
```

Kode tersebut mengimplementasikan Doubly Linked List dengan fungsi untuk menambahkan elemen di awal (**insertFirst**) dan di akhir (**insertLast**) list. Struktur **Node** memiliki data, pointer **next** ke node berikutnya, dan pointer **prev** ke node sebelumnya. Fungsi **insertFirst** menempatkan elemen baru di depan, memperbarui **head**, sementara **insertLast** menambah elemen di akhir dan memperbarui **tail**. Fungsi **printList** mencetak semua elemen dari depan ke belakang dengan format tertentu, memisahkan tiap elemen dengan tanda **<->**. Program menerima tiga input dari pengguna untuk menguji fungsi penambahan di awal dan akhir, lalu menampilkan seluruh isi list.

OUTPUT:

```
• * Executing task: C:/Windows/System32/cmd.exe /d /c .\build\Debug\outDebug.exe  
DAFTAR ANGGOTA LIST: 25 <-> 40 <-> 60  
* Terminal will be reused by tasks, press any key to close it.
```

2.

```
1 #include <iostream>
2 using namespace std;
3
4 // Struktur Node untuk Doubly Linked List
5 struct Node {
6     int data;
7     Node* next;
8     Node* prev;
9 };
10
11 // Fungsi untuk menambahkan elemen di akhir list
12 void insertLast(Node*& head, int value) {
13     Node* newNode = new Node();
14     newNode->data = value;
15     newNode->next = nullptr;
16
17     if (head == nullptr) {
18         newNode->prev = nullptr;
19         head = newNode;
20     } else {
21         Node* temp = head;
22         while (temp->next != nullptr) {
23             temp = temp->next;
24         }
25         temp->next = newNode;
26         newNode->prev = temp;
27     }
28 }
29
30 // Fungsi untuk menghapus elemen pertama dalam list
31 void deleteFirst(Node*& head) {
32     if (head == nullptr) {
33         cout << "List kosong, tidak ada elemen untuk dihapus." << endl;
34         return;
35     }
36
37     Node* temp = head;
38     head = head->next;
39
40     if (head != nullptr) {
41         head->prev = nullptr;
42     }
43
44     delete temp;
45 }
46
47 // Fungsi untuk menghapus elemen terakhir dalam list
48 void deleteLast(Node*& head) {
49     if (head == nullptr) {
50         cout << "List kosong, tidak ada elemen untuk dihapus." << endl;
51         return;
52     }
53
54     if (head->next == nullptr) {
55         // Jika hanya ada satu elemen
56         delete head;
57         head = nullptr;
58     } else {
59         Node* temp = head;
60         while (temp->next != nullptr) {
61             temp = temp->next;
62         }
63
64         temp->prev->next = nullptr;
65         delete temp;
66     }
67 }
68
69 // Fungsi untuk menampilkan seluruh elemen dalam list
70 void displayList(Node* head) {
71     if (head == nullptr) {
72         cout << "List kosong." << endl;
73         return;
74     }
75
76     Node* temp = head;
77     cout << "DAFTAR ANGGOTA LIST: ";
78     while (temp != nullptr) {
79         cout << temp->data << " ";
80         temp = temp->next;
81     }
82     cout << endl;
83 }
84
85 int main() {
86     Node* head = nullptr;
87
88     // Input elemen sesuai instruksi
89     insertLast(head, 30); // Elemen pertama
90     insertLast(head, 45); // Elemen kedua
91     insertLast(head, 60); // Elemen ketiga
92
93     cout << "List sebelum penghapusan:" << endl;
94     displayList(head);
95
96     // Menghapus elemen pertama dan terakhir
97     deleteFirst(head);
98     deleteLast(head);
99
100    cout << "List setelah penghapusan:" << endl;
101    displayList(head);
102
103    return 0;
104 }
```

Dalam Doubly Linked List (DLL), setiap node memiliki data serta pointer **next** dan **prev** untuk menunjuk ke node berikutnya dan sebelumnya. Fungsi **insertLast** menambah elemen baru di akhir list; jika DLL kosong, elemen menjadi **head**. Fungsi **deleteFirst** menghapus elemen pertama, dan jika hanya ada satu elemen, **head** diset menjadi **nullptr**. Fungsi **deleteLast** menghapus elemen terakhir dan mengosongkan list jika elemen tersisa hanya satu. Fungsi **displayList** menampilkan semua elemen dalam DLL. Program utama menambahkan tiga elemen (30, 45, dan 60) lalu menampilkan DLL sebelum dan sesudah penghapusan elemen pertama dan terakhir.

OUTPUT:

```
* Executing task: C:/windows/System32/cmd.exe /d /c .\build\Debug\outDebug.exe

List sebelum penghapusan:
DAFTAR ANGGOTA LIST: 30 45 60
List setelah penghapusan:
DAFTAR ANGGOTA LIST: 45
* Terminal will be reused by tasks, press any key to close it.
```

3.

```
1 #include <iostream>
2 using namespace std;
3
4 // Struktur Node untuk Doubly Linked List
5 struct Node {
6     int data;
7     Node* next;
8     Node* prev;
9 };
10
11 // Fungsi untuk menambahkan elemen di akhir list
12 void insertLast(Node*& head, int value) {
13     Node* newNode = new Node();
14     newNode->data = value;
15     newNode->next = nullptr;
16
17     if (head == nullptr) {
18         newNode->prev = nullptr;
19         head = newNode;
20     } else {
21         Node* temp = head;
22         while (temp->next != nullptr) {
23             temp = temp->next;
24         }
25         temp->next = newNode;
26         newNode->prev = temp;
27     }
28 }
29
30 // Fungsi untuk menampilkan elemen dari depan ke belakang
31 void displayForward(Node* head) {
32     if (head == nullptr) {
33         cout << "List kosong." << endl;
34         return;
35     }
36
37     Node* temp = head;
38     cout << "Elemen dari depan ke belakang: ";
39     while (temp != nullptr) {
40         cout << temp->data << " ";
41         temp = temp->next;
42     }
43     cout << endl;
44 }
45
46 // Fungsi untuk menampilkan elemen dari belakang ke depan
47 void displayBackward(Node* head) {
48     if (head == nullptr) {
49         cout << "List kosong." << endl;
50         return;
51     }
52
53     Node* temp = head;
54
55     // Bergerak ke elemen terakhir
56     while (temp->next != nullptr) {
57         temp = temp->next;
58     }
59
60     cout << "Elemen dari belakang ke depan: ";
61     // Menampilkan dari elemen terakhir ke awal
62     while (temp != nullptr) {
63         cout << temp->data << " ";
64         temp = temp->prev;
65     }
66     cout << endl;
67 }
68
69 int main() {
70     Node* head = nullptr;
71
72     // Menambahkan 4 elemen ke dalam list
73     insertLast(head, 10); // Elemen pertama
74     insertLast(head, 20); // Elemen kedua
75     insertLast(head, 30); // Elemen ketiga
76     insertLast(head, 40); // Elemen keempat
77
78     // Menampilkan elemen dari depan ke belakang
79     displayForward(head);
80
81     // Menampilkan elemen dari belakang ke depan
82     displayBackward(head);
83
84     return 0;
85 }
86
```

Program DLL ini memiliki node dengan tiga komponen: **data** untuk menyimpan nilai elemen, **next** sebagai pointer ke node berikutnya, dan **prev** sebagai pointer ke node sebelumnya. Fungsi **insertLast** menambahkan elemen baru di akhir list, di mana elemen pertama menjadi **head** jika list kosong, dan jika tidak, elemen baru ditambahkan di akhir. Fungsi **displayForward** menampilkan semua elemen dari depan ke belakang dengan memulai dari **head** dan melanjutkan ke elemen berikutnya melalui **next**, sedangkan **displayBackward** menampilkan elemen dari belakang ke depan, mencari node terakhir terlebih dahulu lalu bergerak mundur menggunakan **prev**. Program utama menambahkan empat elemen (10, 20, 30, dan 40) ke dalam list, lalu menampilkan elemen-elemen tersebut dalam dua arah: dari depan ke belakang dan dari belakang ke depan.

OUTPUT:

```
● * Executing task: C:/windows/System32/cmd.exe /d /c .\build\Debug\outDebug.exe

Elemen dari depan ke belakang: 10 20 30 40
Elemen dari belakang ke depan: 40 30 20 10
* Terminal will be reused by tasks, press any key to close it.
```


UNGUIDED

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 // Struktur Node untuk menyimpan data buku
7 struct Node {
8     int IDBuku;
9     string JudulBuku;
10    string PenulisBuku;
11    Node* next;
12    Node* prev;
13 };
14
15 // Fungsi untuk membuat node baru
16 Node* newNode(int IDBuku, string JudulBuku, string PenulisBuku) {
17     Node* node = new Node;
18     node->IDBuku = IDBuku;
19     node->JudulBuku = JudulBuku;
20     node->PenulisBuku = PenulisBuku;
21     node->next = nullptr;
22     node->prev = nullptr;
23     return node;
24 }
25
26 // Fungsi untuk menambahkan buku di akhir linked list
27 void append(Node** head, Node** tail, int IDBuku, string JudulBuku, string PenulisBuku) {
28     Node* new_node = newNode(IDBuku, JudulBuku, PenulisBuku);
29
30     // Jika list kosong
31     if (*head == nullptr) {
32         *head = new_node;
33         *tail = new_node;
34         return;
35     }
36
37     // Tambahkan node baru di akhir list
38     (*tail)->next = new_node;
39     new_node->prev = *tail;
40     *tail = new_node;
41 }
42
43 // Fungsi untuk menampilkan semua buku dari awal ke akhir
44 void printList(Node* node) {
45     while (node != nullptr) {
46         cout << "ID Buku: " << node->IDBuku << endl;
47         cout << "Judul Buku: " << node->JudulBuku << endl;
48         cout << "Penulis Buku: " << node->PenulisBuku << endl;
49         cout << endl;
50         node = node->next;
51     }
52 }
53
54 // Fungsi untuk menampilkan semua buku dari akhir ke awal
55 void printReverseList(Node* node) {
56     while (node != nullptr) {
57         cout << "ID Buku: " << node->IDBuku << endl;
58         cout << "Judul Buku: " << node->JudulBuku << endl;
59         cout << "Penulis Buku: " << node->PenulisBuku << endl;
60         cout << endl;
61         node = node->prev;
62     }
63 }
64
65 int main() {
66     Node* head = nullptr;
67     Node* tail = nullptr;
68
69     // Menambahkan beberapa buku ke dalam linked list
70     append(&head, &tail, 1, "Art of War", "Sun Tzu");
71     append(&head, &tail, 2, "Shrek", "William Steig");
72     append(&head, &tail, 3, "I Have No Mouth, and I Must Scream", "Harlan Ellison");
73
74     cout << "Daftar Buku dari awal ke akhir:" << endl;
75     printList(head);
76
77     cout << "Daftar Buku dari akhir ke awal:" << endl;
78     printReverseList(tail);
79
80     return 0;
81 }
82
```

Kode di atas adalah implementasi linked list ganda untuk menyimpan data buku, di mana setiap node menyimpan ID, judul, dan penulis buku. Fungsi **newNode** membuat node baru, sedangkan **append** menambahkan node di akhir list dengan memperbarui pointer **next** dan **prev**. Fungsi **printList** mencetak daftar buku dari awal hingga akhir, sementara **printReverseList** mencetaknya dari akhir ke awal. Di dalam **main**, list diinisialisasi dengan **head** dan **tail** bernilai **nullptr**, kemudian beberapa buku ditambahkan menggunakan **append**, dan akhirnya daftar buku ditambahkan dalam kedua arah dengan memanggil **printList** dan **printReverseList**.

OUTPUT:

```
● * Executing task: C:/windows/System32/cmd.exe /d /c .\build\Debug\outDebug.exe

Daftar buku dari awal ke akhir:
ID Buku: 101, Judul Buku: Pemrograman C++, Penulis: John Doe
ID Buku: 102, Judul Buku: Struktur Data, Penulis: Jane Smith
ID Buku: 103, Judul Buku: Algoritma Pemrograman, Penulis: Albert Johnson
Daftar buku dari akhir ke awal:
ID Buku: 103, Judul Buku: Algoritma Pemrograman, Penulis: Albert Johnson
ID Buku: 102, Judul Buku: Struktur Data, Penulis: Jane Smith
ID Buku: 101, Judul Buku: Pemrograman C++, Penulis: John Doe
* Terminal will be reused by tasks, press any key to close it.
```

GUIDED

```
1 #include <iostream>
2 using namespace std;
3
4 class Node {
5 public:
6     int data;
7     Node* prev;
8     Node* next;
9 };
10
11 class DoublyLinkedList {
12 public:
13     Node* head;
14     Node* tail;
15
16     // Constructor untuk inisialisasi head dan tail
17     DoublyLinkedList() {
18         head = nullptr;
19         tail = nullptr;
20     }
21
22     // Fungsi untuk menambahkan elemen di depan list (push)
23     void push(int data) {
24         Node* newnode = new Node;
25         newnode->data = data;
26         newnode->prev = nullptr;
27         newnode->next = head;
28
29         if (head != nullptr) {
30             head->prev = newnode;
31         } else {
32             tail = newnode; // Jika list kosong, tail juga mengarah ke node baru
33         }
34         head = newnode;
35     }
36
37     // Fungsi untuk menghapus elemen dari depan list (pop)
38     void pop() {
39         if (head == nullptr) {
40             return; // Jika list kosong
41         }
42         Node* temp = head;
43         head = head->next;
44         if (head != nullptr) {
45             head->prev = nullptr;
46         } else {
47             tail = nullptr; // Jika hanya satu elemen di list
48         }
49         delete temp; // Hapus elemen
50     }
51
52     // Fungsi untuk mengupdate data di list
53     bool update(int oldData, int newData) {
54         Node* current = head;
55         while (current != nullptr) {
56             if (current->data == oldData) {
57                 current->data = newData;
58                 return true; // Jika data ditemukan dan diupdate
59             }
60             current = current->next;
61         }
62         return false; // Jika data tidak ditemukan
63     }
64
65     // Fungsi untuk menghapus semua elemen di list
66     void deleteAll() {
67         Node* current = head;
68         while (current != nullptr) {
69             Node* temp = current;
70             current = current->next;
71             delete temp;
72         }
73         head = nullptr;
74         tail = nullptr;
75     }
76
77     // Fungsi untuk menampilkan semua elemen di list
78     void display() {
79         Node* current = head;
80         while (current != nullptr) {
81             cout << current->data << " ";
82             current = current->next;
83         }
84         cout << endl;
85     }
86 };
87
88 int main() {
89     DoublyLinkedList list;
90     while (true) {
91         cout << "1. Add data" << endl;
92         cout << "2. Delete data" << endl;
93         cout << "3. Update data" << endl;
94         cout << "4. Clear data" << endl;
95         cout << "5. Display data" << endl;
96         cout << "6. Exit" << endl;
97
98         int choice;
99         cout << "Enter your choice: ";
100         cin >> choice;
101
102         switch (choice) {
103             case 1: {
104                 int data;
105                 cout << "Enter data to add: ";
106                 cin >> data;
107                 list.push(data);
108                 break;
109             }
110             case 2: {
111                 list.pop();
112                 break;
113             }
114             case 3: {
115                 int oldData, newData;
116                 cout << "Enter old data: ";
117                 cin >> oldData;
118                 cout << "Enter new data: ";
119                 cin >> newData;
120                 bool updated = list.update(oldData, newData);
121                 if (updated) {
122                     cout << "Data not found" << endl;
123                 }
124                 break;
125             }
126             case 4: {
127                 list.deleteAll();
128                 break;
129             }
130             case 5: {
131                 list.display();
132                 break;
133             }
134             case 6: {
135                 return 0;
136             }
137             default: {
138                 cout << "Invalid choice" << endl;
139                 break;
140             }
141         }
142     }
143     return 0;
144 }
```