LAPORAN PRAKTIKUM

MODUL 6

"DOUBLE LINKED LIST (BAGIAN PERTAMA)"

Disusun Oleh:

MARSELLA DWI JULIANTI (2311104004)

SE 07-01

Dosen :

Yudha Islami Sulistya, S.kom, M.Cs.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

SOAL TP
Soal 1: Menambahkan Elemen di Awal dan Akhir DLL

Deskripsi Soal:
Buatlah program yang mengizinkan pengguna menambahkan elemen ke dalam Doubly Linked List di awal dan di akhir list.

Instruksi:
1. Implementasikan fungsi `insertFirst` untuk menambahkan elemen di awal list.
2. Implementasikan fungsi `insertLast` untuk menambahkan elemen di akhir list.
3. Tampilkan seluruh elemen dalam list dari depan ke belakang setelah penambahan dilakukan.

Contoh Input:
- Input: Masukkan elemen pertama = 10
- Input: Masukkan elemen kedua di awal = 5
- Input: Masukkan elemen ketiga di akhir = 20

Output:
- DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20

Jawab:

Inputan file SOAL01.cpp:

guided01.cpp | UnguidedMod6.cpp | SOAL01.cpp

TP > SOAL01.cpp > main()

```c
#include <stdio.h>
#include <stdlib.h>

// Struktur Node Doubly Linked List
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;

// Struktur Doubly Linked List
typedef struct {
    Node* head;
    Node* tail;
} DoublyLinkedList;

// Fungsi untuk membuat node baru
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

// Fungsi untuk menambahkan elemen di awal list
void insertFirst(DoublyLinkedList* list, int data) {
    Node* newNode = createNode(data);
    if (list->head == NULL) {
        list->head = newNode;
        list->tail = newNode;
    } else {
        newNode->next = list->head;
        list->head->prev = newNode;
        list->head = newNode;
    }
}

// Fungsi untuk menambahkan elemen di akhir list
void insertLast(DoublyLinkedList* list, int data) {
    Node* newNode = createNode(data);
    if (list->tail == NULL) {
        list->head = newNode;
        list->tail = newNode;
    } else {
        newNode->prev = list->tail;
        list->tail->next = newNode;
        list->tail = newNode;
    }
}

// Fungsi untuk menampilkan elemen dalam list dari depan ke belakang
void printList(DoublyLinkedList* list) {
    Node* temp = list->head;
    printf("DAFTAR ANGGOTA LIST: ");
    while (temp != NULL) {
        printf("%d", temp->data);
        if (temp->next != NULL) {
            printf(" <-> ");
```

```
            printf(" <-> ");
        }
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    DoublyLinkedList list;
    list.head = NULL;
    list.tail = NULL;

    int elemen1, elemen2, elemen3;
    printf("Input: Masukkan elemen pertama = ");
    scanf("%d", &elemen1);
    insertLast(&list, elemen1);

    printf("Input: Masukkan elemen kedua di awal = ");
    scanf("%d", &elemen2);
    insertFirst(&list, elemen2);

    printf("Input: Masukkan elemen ketiga di akhir = ");
    scanf("%d", &elemen3);
    insertLast(&list, elemen3);

    printList(&list);

    return 0;
}
```

Pada program diatas, akan menambahkan elemen di awal dan akhir list menggunakan fungsi '**insertFirst**' dan '**insertLast**', kemudian menampilkan elemen dalam list dari depan ke belakang menggunakan fungsi '**printList**'.

Output program:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\06_Double-Linked_List_Bagian_1> cd 'd:\SEMES
R DATA\06_Double-Linked_List_Bagian_1\TP\output'
PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\06_Double-Linked_List_Bagian_1\TP\output> &
Input: Masukkan elemen pertama = 10
Input: Masukkan elemen kedua di awal = 5
Input: Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20
PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\06_Double-Linked_List_Bagian_1\TP\output>
```

Soal 2: Menghapus Elemen di Awal dan Akhir DLL

Deskripsi Soal:
Buatlah program yang memungkinkan pengguna untuk menghapus elemen pertama dan elemen terakhir dalam Doubly Linked List.

Instruksi:

1. Implementasikan fungsi `deleteFirst` untuk menghapus elemen pertama.
2. Implementasikan fungsi `deleteLast` untuk menghapus elemen terakhir.
3. Tampilkan seluruh elemen dalam list setelah penghapusan dilakukan.

Contoh Input:

- Input: Masukkan elemen pertama = 10
- Input: Masukkan elemen kedua di akhir = 15
- Input: Masukkan elemen ketiga di akhir = 20
- Hapus elemen pertama dan terakhir.

Output:

- DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: 15

Jawab:

Inputan file SOAL02.cpp:

TP > SOAL02.cpp > main()

```cpp
#include <stdio.h>
#include <stdlib.h>

// Struktur Node Doubly Linked List
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;

// Struktur Doubly Linked List
typedef struct {
    Node* head;
    Node* tail;
} DoublyLinkedList;

// Fungsi untuk membuat node baru
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

// Fungsi untuk menambahkan elemen di akhir list
void insertLast(DoublyLinkedList* list, int data) {
    Node* newNode = createNode(data);
    if (list->tail == NULL) {
        list->head = newNode;
        list->tail = newNode;
    } else {
        newNode->prev = list->tail;
        list->tail->next = newNode;
        list->tail = newNode;
    }
}

// Fungsi untuk menghapus elemen pertama
void deleteFirst(DoublyLinkedList* list) {
    if (list->head == NULL) {
        printf("List is empty, nothing to delete.\n");
        return;
    }
    Node* temp = list->head;
    list->head = list->head->next;
    if (list->head != NULL) {
        list->head->prev = NULL;
    } else {
        list->tail = NULL; // List menjadi kosong
    }
    free(temp);
}

// Fungsi untuk menghapus elemen terakhir
void deleteLast(DoublyLinkedList* list) {
    if (list->tail == NULL) {
        printf("List is empty, nothing to delete.\n");
```

```
63          return;
64      }
65      Node* temp = list->tail;
66      list->tail = list->tail->prev;
67      if (list->tail != NULL) {
68          list->tail->next = NULL;
69      } else {
70          list->head = NULL; // List menjadi kosong
71      }
72      free(temp);
73  }
74
75  // Fungsi untuk menampilkan elemen dalam list
76  void printList(DoublyLinkedList* list) {
77      Node* temp = list->head;
78      printf("DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: ");
79      while (temp != NULL) {
80          printf("%d", temp->data);
81          if (temp->next != NULL) {
82              printf(" <-> ");
83          }
84          temp = temp->next;
85      }
86      printf("\n");
87  }
88
89  int main() {
90      DoublyLinkedList list;
91      list.head = NULL;
```

```
92      list.tail = NULL;
93
94      int elemen1, elemen2, elemen3;
95      printf("Input: Masukkan elemen pertama = ");
96      scanf("%d", &elemen1);
97      insertLast(&list, elemen1);
98
99      printf("Input: Masukkan elemen kedua di akhir = ");
100     scanf("%d", &elemen2);
101     insertLast(&list, elemen2);
102
103     printf("Input: Masukkan elemen ketiga di akhir = ");
104     scanf("%d", &elemen3);
105     insertLast(&list, elemen3);
106
107     // Menghapus elemen pertama dan terakhir
108     deleteFirst(&list);
109     deleteLast(&list);
110
111     printList(&list);
112
113     return 0;
114 }
```

Pada program diatas, Semua fungsi yang disebutkan di atas akan melakukan hal-hal seperti Fungsi ‘insertLast’ memungkinkan pengguna memasukkan tiga elemen ke dalam List Ganda Terhubung, Fungsi ‘deleteFirst’ menghapus elemen pertama, dan Fungsi ‘deleteLast’ menghapus elemen terakhir; dan Fungsi ‘printList’ menampilkan elemen yang tersisa dalam list setelah penghapusan.

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\06_Double-Linked_List_Bagian_1> cd 'd:\SEMEST
R DATA\06_Double-Linked_List_Bagian_1\TP\output'
PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\06_Double-Linked_List_Bagian_1\TP\output> & .
Input: Masukkan elemen pertama = 10
Input: Masukkan elemen kedua di akhir = 15
Input: Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: 15
PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\06_Double-Linked_List_Bagian_1\TP\output>
```

Soal 3: Menampilkan Elemen dari Depan ke Belakang dan Sebaliknya

Deskripsi Soal: Buatlah program yang memungkinkan pengguna memasukkan beberapa elemen ke dalam Doubly Linked List. Setelah elemen dimasukkan, tampilkan seluruh elemen dalam list dari depan ke belakang, kemudian dari belakang ke depan.

Instruksi:

1. Implementasikan fungsi untuk menampilkan elemen dari depan ke belakang.
2. Implementasikan fungsi untuk menampilkan elemen dari belakang ke depan.
3. Tambahkan 4 elemen ke dalam list dan tampilkan elemen tersebut dalam dua arah.

Contoh Input:

- Input: Masukkan 4 elemen secara berurutan: 1, 2, 3, 4

Output:

- Daftar elemen dari depan ke belakang: 1 <-> 2 <-> 3 <-> 4
- Daftar elemen dari belakang ke depan: 4 <-> 3 <-> 2 <-> 1

Jawab:

Inputan file SOAL03.cpp:

guided01.cpp | UnguidedMod6.cpp | SOAL01.cpp

TP > SOAL03.cpp > main()

```c
#include <stdio.h>
#include <stdlib.h>

// Struktur Node Doubly Linked List
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;

// Struktur Doubly Linked List
typedef struct {
    Node* head;
    Node* tail;
} DoublyLinkedList;

// Fungsi untuk membuat node baru
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

// Fungsi untuk menambahkan elemen di akhir list
void insertLast(DoublyLinkedList* list, int data) {
    Node* newNode = createNode(data);
```

```c
    if (list->tail == NULL) {
        list->head = newNode;
        list->tail = newNode;
    } else {
        newNode->prev = list->tail;
        list->tail->next = newNode;
        list->tail = newNode;
    }
}

// Fungsi untuk menampilkan elemen dalam list dari depan ke belakang
void printListForward(DoublyLinkedList* list) {
    Node* temp = list->head;
    printf("Daftar elemen dari depan ke belakang: ");
    while (temp != NULL) {
        printf("%d", temp->data);
        if (temp->next != NULL) {
            printf(" <-> ");
        }
        temp = temp->next;
    }
    printf("\n");
}

// Fungsi untuk menampilkan elemen dalam list dari belakang ke depan
void printListBackward(DoublyLinkedList* list) {
    Node* temp = list->tail;
    printf("Daftar elemen dari belakang ke depan: ");
    while (temp != NULL) {
```

```c
        printf("%d", temp->data);
        if (temp->prev != NULL) {
            printf(" <-> ");
        }
        temp = temp->prev;
    }
    printf("\n");
}

int main() {
    DoublyLinkedList list;
    list.head = NULL;
    list.tail = NULL;

    int elemen;
    // Menambahkan 4 elemen ke dalam list
    for (int i = 0; i < 4; i++) {
        printf("Input: Masukkan elemen ke-%d = ", i + 1);
        scanf("%d", &elemen);
        insertLast(&list, elemen);
    }

    // Menampilkan elemen dari depan ke belakang
    printListForward(&list);
    // Menampilkan elemen dari belakang ke depan
    printListBackward(&list);

    return 0;
}
```

Pada program diatas, terdapat:

1. Struktur Node dan Doubly Linked List, dimana Program ini menggunakan struktur ‘Node’ untuk menyimpan pointer dan data ke node sebelumnya dan berikutnya, dan struktur ‘Doubly LinkedList’ menyimpan pointer ke head dan tail dari list.
2. Fungsi ‘createNode’ untuk Memungkinkan program untuk membuat node baru dengan data yang diberikan.
3. Fungsi ‘insertLast’ untuk memungkinkan penambahan elemen baru di akhir list.
4. Fungsi ‘printListForward’ untuk memungkinkan penampilan elemen dalam list dari depan ke belakang.
5. Fungsi ‘printListBackward’ untuk memungkinkan penampilan elemen dalam list dari belakang ke depan.
6. Fungsi ‘main’ untuk mengaktifkan dua daftar terhubung, meminta pengguna untuk memasukkan empat elemen secara berurutan dan memanggil pilihan untuk menampilkan elemen dari belakang ke depan dan dari depan ke belakang.

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\06_Double-Linked_List_Bagian_1> cd 'd:\SEMEST
_1\TP\output'
PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\06_Double-Linked_List_Bagian_1\TP\output> &
Input: Masukkan elemen ke-1 = 1
Input: Masukkan elemen ke-2 = 2
Input: Masukkan elemen ke-3 = 3
Input: Masukkan elemen ke-4 = 4
Daftar elemen dari depan ke belakang: 1 <-> 2 <-> 3 <-> 4
Daftar elemen dari belakang ke depan: 4 <-> 3 <-> 2 <-> 1
PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\06_Double-Linked_List_Bagian_1\TP\output>
```