Tugas 4
Praktikum Struktur Data
“Doubly Linked List”

Dosen Pengampu :

Drs. Denny Kurniadi, M.Kom

Disusun oleh

Nama	**: Carel Habsian Osagi**
Nim	**: 23343061**

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2024

Insertion at front:

Nomor	Baris Program	Petikan Source Code	Penjelasan
1.	13-27	`struct Node { int data; struct Node *next; // Pointer to next node struct Node *prev; // Pointer to previous node };`	Deklarasi struktur baru dengan nama `Node` (simpul). Pointer `next` dan `prev` digunakan untuk mengarahkan ke simpul sebelum atau setelah simpul baru yang dibuat.
2.	29-33	`void push(struct Node** head_ref, int new_data) { /* 1. allocate node */ struct Node* new_node = (struct Node*)malloc(sizeof(struct Node)); /* 2. put in the data */ new_node->data = new_data; /* 3. Make next of new node as head and previous as NULL */ new_node->next = (*head_ref); new_node->prev = NULL; /* 4. change prev of head node to new node */ if ((*head_ref) != NULL) (*head_ref)->prev = new_node; /* 5. move the head to point to the new node */ (*head_ref) = new_node; }`	Fungsi `push` digunakan untuk memasukkan elemen baru di depan linked list.
3.	35-48	`void printList(struct Node* node) { struct Node* last; printf("\nTraversal in forward direction \n"); while (node != NULL) { printf(" %d ", node->data); last = node; node = node->next; } printf("\nTraversal in reverse direction \n");`	Fungsi `printList` digunakan untuk mencetak isi linked list secara berurutan dan terbalik.

		`    while (last != NULL) {`
		`    printf(" %d ", last->data);`
		`    last = last->prev;`
		`    }`
		`}`	

4.	50-64	`int main()`
		`{`
		`    /* Start with the empty list */`
		`    struct Node* head = NULL;`
		`    push(&head, 6);`
		`    push(&head, 5);`
		`    push(&head, 2);`
		`    printf("Created DLL is: ");`
		`    printList(head);`
		`    getchar();`
		`    return 0;`
		`}`
			Program utama yang membuat linked list kosong, memanggil fungsi `push` untuk menambahkan elemen, dan mencetak isi linked list.

Insertion After given Node:

Nomor	**Baris Program**	**Petikan Source Code**	**Penjelasan**

1.	16-30	`struct Node`
		`{`
		`    int data;`
		`    struct Node* next; // Pointer to next node`
		`    struct Node* prev; // Pointer to previous node`
		`};`
			Deklarasi struktur baru dengan nama `Node` (simpul). Pointer `next` dan `prev` digunakan untuk mengarahkan ke simpul sebelum atau setelah simpul baru yang dibuat.

2.	32-39	`void push(struct Node** head_ref, int new_data)`
		`{`
		`    /* 1. allocate node */`
		`    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));`
		
		`    /* 2. put in the data */`
		`    new_node->data = new_data;`
		
		`    /* 3. Make next of new node as head and previous as NULL */`
		`    new_node->next = (*head_ref);`
		`    new_node->prev = NULL;`
			Fungsi `push` digunakan untuk memasukkan elemen baru di depan linked list.

		`/* 4. change prev of head node to new node */`
		`if ((*head_ref) != NULL)`
		`(*head_ref)->prev = new_node;`
		
		`/* 5. move the head to point to the new node */`
		`(*head_ref) = new_node;`
		`}`	

3.	41-61	`void insertAfter(struct Node* prev_node, int new_data)`
		`{`
		`/*1. check if the given prev_node is NULL */`
		`if (prev_node == NULL)`
		`{`
		`printf("the given previous node cannot be NULL");`
		`return;`
		`}`
		`/* 2. allocate new node */`
		`struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));`
		`/* 3. put in the data */`
		`new_node->data = new_data;`
		`/* 4. Make next of new node as next of prev_node */`
		`new_node->next = prev_node->next;`
		`/* 5. Make the next of prev_node as new_node */`
		`prev_node->next = new_node;`
		`/* 6. Make prev_node as previous of new_node */`
		`new_node->prev = prev_node;`
		`/* 7. Change previous of new_node's next node */`
		`if (new_node->next != NULL)`
			Fungsi `insertAfter` digunakan untuk memasukkan elemen baru setelah simpul yang ditentukan dalam linked list.

		`        new_node->next->prev = new_node;`
		`}`	

4.	63-80	`void printList(struct Node* node)`
		`{`
		`    struct Node* last;`
		`    printf("\nTraversal in forward direction \n");`
		`    while (node != NULL)`
		`    {`
		`        printf(" %d ", node->data);`
		`        last = node;`
		`        node = node->next;`
		`    }`
		`    printf("\nTraversal in reverse direction \n");`
		`    while (last != NULL)`
		`    {`
		`        printf(" %d ", last->data);`
		`        last = last->prev;`
		`    }`
		`}`
			Fungsi `printList` digunakan untuk mencetak isi linked list secara berurutan dan terbalik.

5.	82-96	`int main()`
		`{`
		`    /* Start with the empty list */`
		`    struct Node* head = NULL;`
		`    push(&head, 6);`
		`    push(&head, 5);`
		`    push(&head, 2);`
		`    insertAfter(head->next, 5);`
		`    printf("Created DLL is: ");`
		`    printList(head);`
		`    getchar();`
		`    return 0;`
		`}`
			Program utama yang membuat linked list kosong, memanggil fungsi `push` dan `insertAfter` untuk menambahkan elemen, dan mencetak isi linked list.

Insertion at End:

Nomor	Baris Program	Petikan Source Code	Penjelasan

1.	16-30	`struct Node`
		`{`
			Deklarasi struktur baru dengan nama `Node` (simpul). Pointer `next` dan `prev`

		`int data; struct Node* next; // Pointer to next node struct Node* prev; // Pointer to previous node };`	digunakan untuk mengarahkan ke simpul sebelum atau setelah simpul baru yang dibuat.
2.	32-39	`void push(struct Node** head_ref, int new_data) { /* 1. allocate node */ struct Node* new_node = (struct Node*)malloc(sizeof(struct Node)); /* 2. put in the data */ new_node->data = new_data; /* 3. Make next of new node as head and previous as NULL */ new_node->next = (*head_ref); new_node->prev = NULL; /* 4. change prev of head node to new node */ if ((*head_ref) != NULL) (*head_ref)->prev = new_node; /* 5. move the head to point to the new node */ (*head_ref) = new_node; }`	Fungsi `push` digunakan untuk memasukkan elemen baru di depan linked list.
3.	41-56	`void append(struct Node** head_ref, int new_data) { /* 1. allocate node */ struct Node* new_node = (struct Node*)malloc(sizeof(struct Node)); struct Node* last = *head_ref; /* used in step 5*/`	Fungsi `append` digunakan untuk memasukkan elemen baru di akhir linked list.

		`/* 2. put in the data */` `new_node->data = new_data;` `/* 3. This new node is going to be the last node, so make next of it as NULL*/` `new_node->next = NULL;` `/* 4. If the Linked List is empty, then make the new node as head */` `if (*head_ref == NULL) {` `new_node->prev = NULL;` `*head_ref = new_node;` `return;` `}` `/* 5. Else traverse till the last node */` `while (last->next != NULL)` `last = last->next;` `/* 6. Change the next of last node */` `last->next = new_node;` `/* 7. Make last node as previous of new node */` `new_node->prev = last;` `return;` `}`	
4.	58-76	`void printList(struct Node* node)` `{` `struct Node* last;` `printf("\nTraversal in forward direction \n");` `while (node != NULL) {` `printf(" %d ", node->data);` `last = node;` `node = node->next;` `}` `printf("\nTraversal in reverse direction \n");`	Fungsi `printList` digunakan untuk mencetak isi linked list secara berurutan dan terbalik.

		`    while (last != NULL) {`
		`        printf(" %d ",`
		`last->data);`
		`        last = last->prev;`
		`    }`
		`}`	

5.	78-94	`int main()`
		`{`
		`    /* Start with the`
		`empty list */`
		`    struct Node* head =`
		`NULL;`
		
		`    // Insert 6. So linked`
		`list becomes 6->NULL`
		`    append(&head, 6);`
		
		`    // Insert 7 at the`
		`beginning. So linked list`
		`becomes 7->6->NULL`
		`    push(&head, 7);`
		
		`    // Insert 1 at the`
		`beginning. So linked list`
		`becomes 1->7->6->NULL`
		`    push(&head, 1);`
		
		`    // Insert 4 at the`
		`end. So linked list`
		`becomes 1->7->6->4->NULL`
		`    append(&head, 4);`
		
		`    printf("Created DLL`
		`is: ");`
		`    printList(head);`
		
		`    getchar();`
		`    return 0;`
		`}`
			Program utama yang membuat linked list kosong, memanggil fungsi `push` dan `append` untuk menambahkan elemen, dan mencetak isi linked list.

Insertion before given node :

Nomor	Baris Program	Petikan Source Code	Penjelasan

1.	14-28	`struct Node {`
		`    int data;`
		`    struct Node* next;`
		`    struct Node* prev;`
		`};`
			Deklarasi struktur baru dengan nama `Node` (simpul). Pointer `next`

2.	30-37	`void push(struct Node** head_ref, int new_data) { struct Node* new_node = (struct Node*)malloc(sizeof(struct Node)); new_node->data = new_data; new_node->next = (*head_ref); new_node->prev = NULL; if ((*head_ref) != NULL) (*head_ref)->prev = new_node; (*head_ref) = new_node; }`	Fungsi `push` digunakan untuk memasukkan elemen baru di depan linked list.
3.	39-52	`void insertBefore(struct Node** head_ref, struct Node* next_node, int new_data) { /*1. check if the given next_node is NULL */ if (next_node == NULL) { printf("the given next node cannot be NULL"); return; } /* 2. allocate new node */ struct Node* new_node = (struct Node*)malloc(sizeof(struct Node)); /* 3. put in the data */ new_node->data = new_data; /* 4. Make prev of new node as prev of next_node */ new_node->prev = next_node->prev; /* 5. Make the prev of next_node as new_node */ next_node->prev = new_node; /* 6. Make next_node as next of new_node */ new_node->next = next_node; /* 7. Change next of new_node's previous node */`	Fungsi `insertBefore` digunakan untuk memasukkan elemen baru sebelum simpul yang ditentukan dalam linked list.

		`    if (new_node->prev != NULL)`
		`    new_node->prev->next = new_node;`
		`    /* 8. If the prev of new_node is NULL, it will be`
		`    the new head node */`
		`    else`
		`    (*head_ref) = new_node;`
		`}`	

4.	54-69	`void printList(struct Node* node)`
		`{`
		`    struct Node* last;`
		`    printf("\nTraversal in forward direction \n");`
		`    while (node != NULL) {`
		`    printf(" %d ", node->data);`
		`    last = node;`
		`    node = node->next;`
		`    }`
		`    printf("\nTraversal in reverse direction \n");`
		`    while (last != NULL) {`
		`    printf(" %d ", last->data);`
		`    last = last->prev;`
		`    }`
		`}`
			Fungsi `printList` digunakan untuk mencetak isi linked list secara berurutan dan terbalik.

5.	71-87	`int main()`
		`{`
		`    /* Start with the empty list */`
		`    struct Node* head = NULL;`
		`    push(&head, 7);`
		`    push(&head, 1);`
		`    push(&head, 4);`
		`    insertBefore(&head, head->next, 8);`
		`    printf("Created DLL is: ");`
		`    printList(head);`
		`    getchar();`
		`    return 0;`
		`}`
			Program utama yang membuat linked list kosong, memanggil fungsi `push` dan `insertBefore` untuk menambahkan elemen, dan mencetak isi linked list.