LAPORAN PRAKTIKUM STRUKTUR DATA

JOBSHEET 4

Doubly Linked List

Oleh :

ALVIN ANUGERAH PRATAMA

(22343019)

Dosen Pengampu :

RANDI PROSKA SANDRA, S.PD., M.SC.

PRODI INFORMATIKA
DEPARTEMEN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2022

Analisis Percobaan 1 (Insertion at front)

No.	Baris Program	Source Code	Penjelasan
1.	3 – 4	`#include <stdio.h>` `#include <stdlib.h>`	Menyertakan dua header file standar yaitu stdio.h dan stdlib.h.

2.	7 – 11	`struct Node{`
		`int data;`
		`struct Node *next; // Pointer to next node`
		`struct Node *prev; // Pointer to previous node`
		`};`
			Mendefinisikan struktur data Node yang memiliki tiga field yaitu data (yang berisi data yang disimpan di dalam node), next (yang merupakan pointer ke node selanjutnya), dan prev (yang merupakan pointer ke node sebelumnya).

3.	13 – 27	`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;`
		`}`
			Mendefinisikan fungsi push yang digunakan untuk menambahkan node baru di awal daftar linked list. Fungsi ini menerima dua parameter yaitu pointer ke pointer head (pointer yang menunjuk ke alamat memori variabel head yang berisi alamat memori node pertama) dan data baru yang akan dimasukkan ke dalam node. Fungsi ini melakukan beberapa hal seperti mengalokasikan memori untuk node baru, menempatkan data baru ke dalam node, mengatur pointer next dan prev pada node baru, serta mengubah pointer head menjadi node baru.

4.	29 – 42	`void printList(struct Node* node){`
		`struct Node* last;`
		`printf("\nTraversal in forward direction \n");`
		`while (node != NULL) {`
			Mendefinisikan fungsi printList yang digunakan untuk mencetak semua elemen dalam linked list. Fungsi ini menerima parameter node (pointer yang menunjuk ke node pertama dalam linked list). Fungsi ini mencetak semua

		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;
		
		}
		
		}
			elemen dalam linked list dari depan ke belakang, kemudian mencetak semua elemen dalam linked list dari belakang ke depan.

5.	44 – 56	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;
		
		}
			Mendefinisikan fungsi main. Fungsi ini memulai dengan daftar linked list kosong, kemudian memanggil fungsi push untuk menambahkan tiga node baru dengan nilai masing-masing 2, 5, dan 6. Akhirnya, fungsi printList dipanggil untuk mencetak semua elemen dalam linked list.

Analisis Percobaan 2 (Insertion after given node)

No.	Baris Program	Source Code	Penjelasan

1.	5 – 9	struct Node{
		
		int data;
		
		struct Node *next; // Pointer to next node
		
		struct Node *prev; // Pointer to previous node
		
		};
			Ini adalah definisi struktur Node yang digunakan dalam linked list. Node memiliki tiga elemen, yaitu data integer dan dua pointer, yaitu next dan prev untuk menunjuk ke node selanjutnya dan sebelumnya dalam linked list.

2.	11 - 24	void push(Node** head_ref, int new_data){ /* 1. allocate node */ Node* new_node = new Node();	Fungsi push digunakan untuk menambahkan node baru di awal linked list. Pertama, fungsi ini mengalokasikan memori untuk node baru, kemudian menetapkan

		/* 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;
		
		}
			nilai data untuk node baru dan mengatur next ke node pertama saat ini. Setelah itu, prev diatur sebagai NULL karena node baru akan menjadi node pertama. Selanjutnya, jika linked list tidak kosong, maka prev dari node pertama saat ini harus diatur sebagai node baru. Akhirnya, node baru ditetapkan sebagai node pertama saat ini.

3.	26 - 46	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;
			Fungsi insertAfter digunakan untuk menyisipkan node baru setelah node tertentu. Pertama, fungsi ini memeriksa apakah node sebelumnya bukan NULL. Jika iya, fungsi akan mengalokasikan memori untuk node baru dan mengatur nilai data dan pointer next untuk node baru. Selanjutnya, pointer next dari node sebelumnya diatur untuk menunjuk ke node baru, dan pointer prev dari node baru diatur untuk menunjuk ke node sebelumnya. Jika ada node setelah node baru, maka pointer prev dari node tersebut harus diatur untuk menunjuk ke node baru.

		/* 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)
		new_node->next->prev = new_node;
		}	

4.	48 - 62	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 linked list dari awal ke akhir dan dari akhir ke awal. Fungsi ini menerima parameter node yang akan dicetak, lalu node terakhir disimpan di variabel last. Fungsi ini melakukan loop selama node tidak NULL dan mencetak nilai data dari setiap node. Kemudian, variabel last diatur ke node saat ini dan loop berakhir saat node NULL. Kemudian, fungsi melakukan loop dari node terakhir ke node pertama dan mencetak nilai data dari setiap node.

5.	64 - 76	int main(){
		/* Start with the empty list */
		struct Node* head = NULL;
		push(&head, 6);
		push(&head, 5);
		push(&head, 2);
		insertAfter(head->next, 5);
			Fungsi main memulai dengan mendefinisikan linked list kosong. Kemudian, tiga node baru ditambahkan menggunakan fungsi push dan satu node baru ditambahkan menggunakan fungsi insertAfter. Akhirnya, linked list dicetak menggunakan fungsi printList.

		printf("Created DLL is: ");
		
		printList(head);
		
		getchar();
		
		return 0;
		
		}	

Analisis Percobaan 3 (Insertion at end)

No.	Baris Program	Source Code	Penjelasan

1.	7 – 11	struct Node{
		
		int data;
		
		struct Node *next; // Pointer to next node
		
		struct Node *prev; // Pointer to previous node
		
		};
			Definisi struct Node tersebut adalah bagian dari implementasi struktur data linked list yang lebih kompleks, yaitu doubly linked list. Dalam doubly linked list, setiap node memiliki dua pointer, yaitu pointer ke node sebelumnya dan pointer ke node selanjutnya, sehingga memungkinkan untuk melakukan traversing maju mundur pada linked list. Hal ini dapat mempermudah proses manipulasi data pada linked list.

2.	13 – 26	void push(Node** head_ref, int new_data){
		
		/* 1. allocate node */
		
		Node* new_node = new 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;
			Fungsi push pada doubly linked list ini berguna untuk menambahkan node baru ke awal linked list (sebagai head node baru). Setiap kali push dilakukan, node baru akan dihubungkan ke head node sebelumnya dan head node sebelumnya diubah pointer prev-nya untuk menunjuk ke node baru. Fungsi ini memudahkan proses pengisian linked list dengan data baru.

		/* 5. move the head to point to the new node */ (*head_ref) = new_node; }	

3.	28 - 53	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*/
		
		/* 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;
			Fungsi append ini digunakan untuk menambahkan sebuah node baru ke akhir dari doubly linked list.
			
			Fungsi ini cocok digunakan ketika ingin menambahkan elemen pada akhir dari list, sehingga dapat menghindari traversal pada seluruh list untuk menemukan node terakhir.

		`return;`
		
		`}`	

4.	55 - 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() berguna untuk mencetak isi dari doubly linked list, baik dari depan ke belakang maupun dari belakang ke depan.
			
			Fungsi ini menggunakan dua loop while, yang pertama untuk mencetak dari depan ke belakang, dan yang kedua untuk mencetak dari belakang ke depan.
			
			Fungsi ini memanfaatkan pointer last untuk menyimpan node terakhir dari linked list, yang kemudian digunakan untuk mencetak isi linked list dari belakang ke depan

5.	71 - 89	`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);`
			Operasi-operasi yang dilakukan pada fungsi main ini dapat menghasilkan linked list dengan urutan node sebagai berikut: 1->7->6->4
			
			Fungsi main ini menunjukkan bagaimana cara menggunakan fungsi-fungsi dasar untuk memanipulasi sebuah doubly linked list, yaitu push, append, dan printList.

		printf("Created DLL is: ");
		
		printList(head);
		
		getchar();
		
		return 0;
		
		}	

Analisis Percobaan 4 (Insertion before given node)

No.	Baris Program	Source Code	Penjelasan

1.	7 – 11	struct Node {
		
		int data;
		
		struct Node* next;
		
		struct Node* prev;
		
		};
			Mendefinisikan struktur Node dengan tiga anggota, yaitu data (int), next (pointer ke Node berikutnya), dan prev (pointer ke Node sebelumnya).

2.	13 – 21	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;
		
		}
			Implementasi fungsi push() yang memasukkan elemen baru ke depan linked list yang ditunjukkan oleh head_ref. Pertama-tama, sebuah node baru dialokasikan dengan malloc(). Kemudian, data diatur pada node baru, next diatur untuk menunjuk ke kepala sebelumnya, dan prev diatur menjadi NULL. Jika kepala sebelumnya bukan NULL, prev kepala sebelumnya diatur untuk menunjuk ke node baru. Akhirnya, head_ref diperbarui untuk menunjuk ke node baru.

3.	23 – 48	/* Given a node as next_node, insert a new node before the given node */
		
		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");
			Implementasi fungsi insertBefore() yang memasukkan elemen baru sebelum node yang diberikan. Pertama-tama, node baru dialokasikan dengan malloc(). Kemudian, data diatur pada node baru, prev diatur untuk menunjuk ke prev dari node berikutnya, next dari node baru diatur untuk menunjuk ke node berikutnya, dan prev dari node berikutnya diatur untuk menunjuk ke node baru. Jika prev dari node baru bukan NULL, next dari node baru yang

		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 */
		
		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;
		
		}
			sebelumnya diatur untuk menunjuk ke node baru. Jika prev dari node baru NULL, head_ref diperbarui untuk menunjuk ke node baru.

4.	50 – 64	void printList(struct Node* node){
		
		struct Node* last;
		
		printf("\nTraversal in forward direction \n");
		
		while (node != NULL) {
		
		printf(" %d ", node->data);
			Mendefinisikan fungsi printlist untuk mencetak isi dari DLL secara terurut maju dan terurut mundur. Fungsi ini menerima satu parameter, yaitu head node dari DLL (node). Pada baris 42, pointer last dideklarasikan untuk menyimpan node terakhir dari DLL. Kemudian, pada baris 44-48,

		last = node;
		node = node->next;
		}
		
		printf("\nTraversal in reverse direction \n");
		while (last != NULL) {
		printf(" %d ", last->data);
		last = last->prev;
		}
		}
			DLL dicetak secara terurut maju menggunakan loop while. Pada setiap iterasinya, data dari setiap node dicetak dan node dipindahkan ke node selanjutnya. Pointer last ditetapkan sebagai node pada setiap iterasi. Kemudian, pada baris 50-54, DLL dicetak secara terurut mundur menggunakan loop while. Pada setiap iterasinya, data dari setiap node dicetak dan last dipindahkan ke node sebelumnya.

5.	66 - 79	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;
		}
			Merupakan fungsi main sebagai program utama yang digunakan untuk menguji fungsi-fungsi yang telah didefinisikan