Jobsheet 4

Praktikum Struktur Data

Dosen pengampu : Randi Proska Sandra, S.Pd, M.Sc

Kode Kelas : 202323430158

Disusun Oleh :

Gema Pratama Mahadi Putera
23343066

PROGRAM STUDI INFORMATIKA (NK)
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2023

1. **Insertion at front**

Nomor Program	Baris Program	Petikan Source Code	Penjelasan

1	6 - 11	`struct Node`
		`{`
		`int data;`
		`struct Node *next;`
		`struct Node *prev;`
		`};`
			Menggunakan struktur data Node untuk merepresentasikan setiap elemen dalam linked list ganda. Setiap `node` memiliki dua pointer, yaitu `next` yang menunjukkan ke node berikutnya, dan `prev` yang menunjukkan ke node sebelumnya.

2	13 - 27	`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;`
		`}`
			Digunakan untuk menambahkan node baru ke depan dari linked list. Ini membuat node baru, menginisialisasi data dan pointer, kemudian mengatur pointer `next` node pertama pada node baru. Jika linked list tidak kosong, node pertama diatur untuk menunjuk ke node baru sebagai node sebelumnya.

3	28 - 42	`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;`
		`}`
		`}`
			Digunakan untuk mencetak isi linked list. Pertama, itu mencetak elemen dalam urutan maju (dari depan ke belakang) dan kemudian mencetak elemen dalam urutan mundur (dari belakang ke depan).

4	43 - 54	`int main()`
		`{`
		`struct Node* head = NULL;`
		`push(&head, 6);`
		`push(&head, 5);`
		`push(&head, 2);`
		`printf("Created DLL is: ");`
		`printList(head);`
		`getchar();`
		`return 0;`
		`}`
			Ini adalah tempat program dimulai. Di sini, beberapa node ditambahkan ke linked list menggunakan fungsi `push()`, kemudian isi linked list dicetak menggunakan fungsi `printList()`.

5	16	`struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));`	Digunakan untuk mengalokasikan memori untuk node baru. Ini memastikan bahwa memori yang cukup dialokasikan untuk menyimpan node baru sebelum data dimasukkan ke dalamnya.

2. Insertion after given node

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

1	7 - 12	`struct Node`
		`{`
		`int data;`
		`struct Node *next;`
		`struct Node *prev;`
		`};`
			Digunakan untuk membuat linked list dua arah (doubly linked list). Setiap node memiliki tiga bagian: data integer dan dua pointer, satu untuk menunjuk ke node berikutnya `(next)` dan yang lainnya untuk menunjuk ke node sebelumnya `(prev)`.

2	13 - 27	`void push(Node** head_ref, int new_data)`
		`{`
		`Node* new_node = new 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;`
		`}`
			Digunakan untuk menambahkan node baru ke depan linked list. Prosesnya mirip dengan sebelumnya, namun disini digunakan operator `new` untuk mengalokasikan memori untuk node baru.

3	28 - 48	`void insertAfter(struct Node* prev_node, int new_data)`
		`{`
		`if (prev_node == NULL) {`
		`printf("the given previous node cannot be NULL");`
		`return;`
		`}`
		`struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));`
		`new_node->data = new_data;`
		`new_node->next = prev_node->next;`
		`prev_node->next = new_node;`
		`new_node->prev = prev_node;`
		`if (new_node->next != NULL)`
		`new_node->next->prev = new_node;`
		`}`
			Memasukkan node baru setelah node tertentu dalam linked list. Ini memerlukan node sebelumnya (`prev_node`) dan data baru. Prosesnya melibatkan penyesuaian pointer `next` dan `prev` untuk menambahkan node baru di antara node yang telah ada.

4	49 - 63	`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;`
		`}`
		`}`
			Tetap sama dengan yang sebelumnya, di mana ia mencetak isi linked list baik dalam urutan maju maupun mundur.

5	64 - 76	`int main()`
		`{`
		`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;`
		`}`
			Juga mirip dengan yang sebelumnya, kecuali bahwa di sini kita menggunakan operator `new` untuk mengalokasikan memori untuk node baru dalam fungsi `push()`. Selain itu, kita menggunakan `insertAfter()` untuk menyisipkan node baru setelah node tertentu.

3. Insertion at end

Nomor Program	Baris Program	Petikan Source Code	Penjelasan

1	6 - 12	`struct Node`
		`{`
		`  int data;`
		`  struct Node *next;`
		`  struct Node *prev;`
		`};`
			Digunakan untuk membuat linked list dua arah (doubly linked list). Setiap node memiliki tiga bagian: data integer dan dua pointer, satu untuk menunjuk ke node berikutnya `(next)` dan yang lainnya untuk menunjuk ke node sebelumnya `(prev)`.

2	13 - 27	`void push(Node** head_ref, int new_data)`
		`  {`
		`  Node* new_node = new 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;`
		`}`
			Digunakan untuk menambahkan node baru ke depan linked list. Prosesnya mirip dengan sebelumnya, namun disini digunakan operator `new` untuk mengalokasikan memori untuk node baru.

3	28 - 53	`void append(struct Node** head_ref, int new_data)`
		`{`
		`  struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));`
		`  struct Node* last = *head_ref; /* used in step 5*/`
		`  new_node->data = new_data;`
		`  new_node->next = NULL;`
		`  if (*head_ref == NULL) {`
		`  new_node->prev = NULL;`
		`  *head_ref = new_node;`
		`  return;`
		`  }`
		`  while (last->next != NULL)`
			Digunakan untuk menambahkan node baru ke akhir linked list. Ini menciptakan node baru, kemudian mencari node terakhir dalam linked list dan menambahkan node baru setelahnya.

		last = last->next;
		last->next = new_node;
		new_node->prev = last;
		return;
		}	

4	54 - 68	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;
		}
		}
			Tetap sama dengan yang sebelumnya, di mana ia mencetak isi linked list baik dalam urutan maju maupun mundur.

5	69 - 88	int main()
		{
		struct Node* head = NULL;
		append(&head, 6);
		push(&head, 7);
		push(&head, 1);
		append(&head, 4);
		printf("Created DLL is: ");
		printList(head);
		getchar();
		return 0;
		}
			Juga mirip dengan yang sebelumnya, di mana kita menggunakan operator `new` untuk mengalokasikan memori untuk node baru dalam fungsi `push()`, dan kita menggunakan `append()` untuk menambahkan node baru ke akhir linked list.

4. Insertion before given node

Nomor Program	Baris Program	Petikan Source Code	Penjelasan
1	7 - 11	**struct Node {** **int data;** **struct Node* next;** **struct Node* prev;** **};**	Digunakan untuk membuat linked list dua arah (doubly linked list). Setiap node memiliki tiga bagian: data integer dan dua pointer, satu untuk menunjuk ke node berikutnya `(next)` dan yang lainnya untuk menunjuk ke node sebelumnya `(prev)`.

2	12 - 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;`
		`}`
			Digunakan untuk menambahkan node baru ke depan linked list. Prosesnya mirip dengan contoh sebelumnya, di mana node baru dibuat, diisi dengan data, dan ditambahkan ke depan linked list.

3	23 - 47	`void insertBefore(struct Node** head_ref, struct Node* next_node, int new_data)`
		`{`
		`if (next_node == NULL) {`
		`printf("the given next node cannot be NULL");`
		`return;`
		`}`
		`struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));`
		`new_node->data = new_data;`
		`new_node->prev = next_node->prev;`
		`next_node->prev = new_node;`
			Memasukkan node baru sebelum node yang ditentukan dalam linked list. Ini memeriksa apakah node berikutnya `(next_node)` tidak NULL, kemudian membuat node baru, mengatur pointer `next` dan `prev`, dan menyesuaikan pointer pada node tetangga yang

		new_node->next = next_node;
		if (new_node->prev != NULL)
		new_node->prev->next = new_node;
		else
		(*head_ref) = new_node;
		}
			terkait.

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;
		}
		}
			Tetap sama seperti yang sebelumnya, mencetak isi linked list baik dalam urutan maju maupun mundur.

5	64 - 76	int main()
		{
		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;
		}
			Juga mirip dengan yang sebelumnya, di mana kita menggunakan `push()` untuk menambahkan beberapa node ke depan linked list, dan kemudian menggunakan `insertBefore()` untuk menyisipkan node baru sebelum node tertentu.