Nama : Oksa Bayu Widrian

NIM : 23343080

Tugas Praktikum Struktur Data 04

Nomor Program	Baris Program	Petikan Source Code	Penjelasan

1.	7-12	`struct Node`
		`{`
		`  int data;`
		`  struct Node *next;`
		`  struct Node *prev;`
		`};`
			Deklarasi struktur baru dengan nama node (simpul). Next dan prev adalah variable pointer yang akan digunakan untuk mengarahkan ke simpul sebelum atau setelah sebuah simpul baru dibuat.

	15-33	`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;`
			Menggunakan fungsi malloc. sizeof(struct Node) digunakan untuk mengalokasikan ruang yang cukup untuk satu instance dari struct Node, dimana struct node ini dibuat untuk membuat data, menyisipkan data ke node baru, Menetapkan Pointer next Node Baru, Menetapkan Pointer prev Node Baru, Mengubah Pointer prev Node Awal (Head), Memindahkan Kepala Daftar ke Node Baru.

	38-45	`struct Node* last;`
		`  printf("\nTraversal in forward direction \n");`
		`  while (node != NULL) {`
		`  printf(" %d ", node->data);`
		`  last = node;`
		`  node = node->next;`
		`  }`
			Node->data menunjuk data yang ada pada node, last=data untuk menyimpan referensi ke node terakhir, dan node=node->next untuk mengggerakkan ‘node’ ke node berikutnya.

	47-51	`printf("\nTraversal in reverse direction \n");`
		`  while (last != NULL) {`
		`  printf(" %d ", last->data);`
		`  last = last->prev`
		`  }`
			Menggunakan loop while untuk mencetak data dari daftar dalam urutan mundur. Dimulai dari node terakhir (yang disimpan dalam variabel `last`), loop akan berjalan sampai `last` menjadi `NULL`, yang menandakan akhir dari daftar.
			Panggilan fungsi push bertujuan untuk menambahkan node baru ke

	61-66	push(&head, 6);
		push(&head, 5);
		push(&head, 2);
		
		printf("Created DLL is: ");
		printList(head);
			awal daftar berantai ganda (head). Setelah node-node tersebut ditambahkan, kita mencetak daftar menggunakan printList(head);. Ini akan mencetak daftar berantai ganda dari kepala hingga ekor (dari node dengan data 2 hingga node dengan data 6) dalam urutan maju dan mundur.

2.	7-12	struct Node
		{
		int data;
		struct Node *next;
		struct Node *prev;
		};
			Deklarasi struktur baru dengan nama node (simpul). Next dan prev adalah variable pointer yang akan digunakan untuk mengarahkan ke simpul sebelum atau setelah sebuah simpul baru dibuat.

	18-32	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;
			Menggunakan fungsi malloc. sizeof(struct Node) digunakan untuk mengalokasikan ruang yang cukup untuk satu instance dari struct Node, dimana struct node ini dibuat untuk membuat data, menyisipkan data ke node baru, Menetapkan Pointer next Node Baru, Menyisipkan Node Baru ke dalam Daftar, Menetapkan Pointer prev Node Baru, dan Mengubah Pointer prev Node Berikutnya.

	45-61	struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
		
		new_node->data = new_data;
		new_node->next = prev_node->next; // Set the next of the new node to the next of the previous node
		prev_node->next = new_node;
		new_node->prev = prev_node;
			Fungsi malloc ini digunakan untuk Pengecekan Node Sebelumnya, Alokasi Memori untuk Node Baru, Menetapkan Data ke dalam Node Baru, Menetapkan Pointer next Node Baru, Menyisipkan Node Baru ke dalam Daftar, Menetapkan Pointer prev Node Baru, Memperbarui Pointer prev Node Berikutnya.

	67-74	`if (new_node->next != NULL)` `new_node->next->prev = new_node;`	Node->data menunjuk data yang ada pada node, last=data untuk menyimpan referensi ke node terakhir, dan node=node->next untuk mengggerakkan 'node' ke node berikutnya.

	76-80	`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`
		`}`
			Menggunakan loop while untuk mencetak data dari daftar dalam urutan mundur. Dimulai dari node terakhir (yang disimpan dalam variabel `last`), loop akan berjalan sampai `last` menjadi `NULL`, yang menandakan akhir dari daftar.

3.	16-27	`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;`
			Pada struct ini digunakan untuk Mengalokasikan Memori untuk Node Baru, Memasukkan Data ke dalam Node Baru, Mengatur Pointer next Node Baru ke Head Lama, Mengatur Pointer prev Node Baru Menjadi NULL, Mengatur Pointer prev Head Lama Menjadi Node Baru, Memindahkan Head ke Node Baru.

	33-43	`struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));`
		
		`struct Node* last = *head_ref;`
		`new_node->data = new_data;`
		`new_node->next = NULL;`
		`if (*head_ref == NULL) {`
		`new_node->prev = NULL;`
			Struct ini berguna untuk Mengalokasikan Memori untuk Node Baru, Menginisialisasi Pointer last untuk Menemukan Node Terakhir, Memasukkan Data ke dalam Node Baru, Mengatur Pointer next Node Baru Menjadi NULL, Mengatur Pointer prev Node Baru

		*head_ref = new_node;	

	47-51	while (last->next != NULL)
		last = last->next;
		last->next = new_node;
		new_node->prev = last;
			Loop ini digunakan untuk mencari node terakhir dalam daftar. Dimulai dari node pertama (last = *head_ref), loop akan terus berjalan selama pointer next dari last tidak menunjuk ke NULL. Ini berarti kita bergerak dari node ke node sampai kita mencapai node terakhir dalam daftar.

	57-70	struct Node* last;
		
		printf("\nTraversal in forward direction: ");
		while (node != NULL) {
		printf("%d ", node->data);
		last = node;
		node = node->next;
		}
		
		printf("\nTraversal in reverse direction: ");
		while (last != NULL) {
		printf("%d ", last->data);
		last = last->prev;
		}
			Pada struct ini ada penulusuran maju Pada bagian ini, kita mulai dari node pertama (node) dan terus bergerak maju sampai menemui node terakhir, Setiap kali kita mencetak data dari node saat ini (node->data), kita juga menyimpan node saat ini dalam variabel last.
			Lalu ada juga penulusuran mundur, dimana menggunakan variabel last yang telah menyimpan referensi ke node terakhir. Kita mulai dari node terakhir dan bergerak mundur sampai ke node pertama.

	74-88	int main() {
		struct Node* head = NULL;
		
		append(&head, 6);
		push(&head, 7);
		push(&head, 1);
		append(&head, 4);
		
		printf("Created Doubly Linked List: ");
		printList(head);
		return 0;
		}
			inisialisasi pointer head untuk menunjuk ke NULL, ini menandakan bahwa daftar berantai ganda saat ini kosong.
			
			Kita menggunakan fungsi append dan push untuk menambahkan elemen ke dalam daftar berantai ganda head.
			
			• append(&head, 6);: Menambahkan node baru dengan nilai 6 ke akhir daftar.
			• push(&head, 7);: Menambahkan node baru

			dengan nilai 7 ke awal daftar. • push(&head, 1);: Menambahkan node baru dengan nilai 1 ke awal daftar. • append(&head, 4);: Menambahkan node baru dengan nilai 4 ke akhir daftar.

4.	14-25	`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;`
			Struct ini berguna untuk Mengalokasikan Memori untuk Node Baru, Menetapkan Data ke dalam Node Baru, Menetapkan Pointer next Node Baru ke Head Sekarang, Menetapkan Pointer prev Node Baru Menjadi NULL, Mengatur Pointer prev Head Sekarang Menjadi Node Baru, Memindahkan Head ke Node Baru.

	31-34	`if (next_node == NULL) {`
		`printf("the given next node cannot be NULL");`
		`return;`
		`}`
			Memeriksa apakah node selanjutnya yang diberikan (next_node) bukan NULL. Jika next_node adalah NULL, itu berarti tidak mungkin untuk menyisipkan node baru sebelumnya. Dalam hal ini, kita mencetak pesan kesalahan dan langsung kembali dari fungsi tanpa melakukan operasi lebih lanjut.

	36-49	`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;`
		`new_node->next = next_node;`
		
		`if (new_node->prev != NULL) new_node->prev->next =`
			Struct ini berguna untuk Mengalokasikan Memori untuk Node Baru, Menetapkan Data ke dalam Node Baru, Menetapkan Pointer prev Node Baru, Menetapkan Pointer prev dari next_node ke Node Baru, Menetapkan Pointer next Node Baru, Mengatur Node Baru Sebagai Kepala Jika Perlu.

		new_node; else (*head_ref) = new_node;	

	54-67	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;
		}
		}
			Pada struct ini ada penulusuran maju Pada bagian ini, kita mulai dari node pertama (node) dan terus bergerak maju sampai menemui node terakhir, Setiap kali kita mencetak data dari node saat ini (node->data), kita juga menyimpan node saat ini dalam variabel last.
			Lalu ada juga penulusuran mundur, dimana menggunakan variabel last yang telah menyimpan referensi ke node terakhir. Kita mulai dari node terakhir dan bergerak mundur sampai ke node pertama.

	71-85	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;
		}
			Pada fungsi int main ini digunakan untuk Inisialisasi Daftar Kosong, Menambahkan Beberapa Elemen ke Daftar, Menyisipkan Node Baru Sebelum Node Kedua, Mencetak Daftar Berantai Ganda, dan Menunggu Pencetan Tombol Sebelum Keluar.