**PRAKTIKUM STRUKTUR DATA**



**DOSEN PEMBIMBING:**

Randi Proska Sandra, S.Pd, M.Sc

**Disusun Oleh:**

Serly Eka Putri

(23343083)

**PRODI INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAR NEGERI PADANG**

**2024**

1. **Insert at front**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nomor**  **Program** | **Baris**  **Program** | **Petikan Source Code** |  |
| **1** | **6-11** | struct Node  {  int data;  structNode \*next;  structNode \*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. |

1. **Insertion after 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** | **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. |

1. **Insert 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)  last = last->next;  last->next = new\_node;  new\_node->prev = last;  return;  } | 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. |
| **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. |

1. **Insert 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;  new\_node->next = next\_node;  if (new\_node->prev != NULL)  new\_node->prev->next = new\_node;  else  (\*head\_ref) = 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 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  } | 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. |