**Praktikum**

**Algoritma Pemrograman**

**Tugas 4**

Dosen Pembimbing:

Randi Proska Sandra, M.Sc



Disusun oleh :

Nama : Rafli Arianto

NIM : 23343051

**UNIVERSITAS NEGERI PADANG**

**FAKULTAS TEKNIK**

**INFORMATIKA**

**2024**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nomor Program** | **Baris Program** | **Petikan Source Code** | **Penjelasan** |
| 1 | 4 - 8 | struct Node{  int data;  struct Node \*next;  struct Node \*prev;  }; | Struct Node untuk membuat double linked list terdiri dari elemen int data untuk menyimpan nilai, dan dua pointer struct Node \*next serta struct Node \*prev untuk menunjuk ke node berikutnya dan sebelumnya dalam list. |
| 1 | 11 - 12 | struct Node \*new\_node = (struct Node\*)malloc(sizeof(struct Node));new\_node->data = new\_data; | Mengalokasikan memori untuk sebuah node baru dari tipe struct Node, kemudian menginisialisasi nilai data dari node tersebut dengan nilai new\_data. |
| 1 | 13 - 14 | new\_node->next = (\*head\_ref);  new\_node->prev = NULL;  if((\*head\_ref) != NULL){  (\*head\_ref)->prev = new\_node;}  (\*head\_ref) = new\_node; | Mengatur node baru (new\_node) sebagai kepala baru dari double linked list dengan membuat next menunjuk ke node lama yang menjadi kepala, mengatur prev dari node baru ke NULL, dan jika kepala lama tidak NULL, mengatur prev dari kepala lama ke node baru, kemudian memperbarui head\_ref agar menunjuk ke node baru tersebut. |
| 1 | 23 - 34 | 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;  } | Melakukan traversal double linked list dari depan ke belakang dengan mencetak nilai data setiap node, kemudian melakukan traversal dari belakang ke depan dengan mencetak nilai data dari node terakhir hingga ke node pertama. |
| 1 | 40 - 45 | struct Node \*head = NULL;  push(&head, 6);  push(&head, 5);  push(&head, 2);  printf("Created DLL is : ");  printList(head); | Menginisialisasi sebuah double linked list kosong, menambahkan tiga node dengan nilai 6, 5, dan 2 di bagian depan list menggunakan fungsi push, dan kemudian mencetak isi double linked list tersebut menggunakan fungsi printList. |
| 2 | 4 – 8 | struct Node{  int data;  struct Node \*next;  struct Node \*prev;  }; struct Node \*new\_node = new Node();  // 2. Memasukkan data  new\_node->data = new\_data; | Mendefinisikan struktur Node untuk double linked list, mengalokasikan memori untuk sebuah node baru menggunakan new Node(), dan menginisialisasi elemen data dari node baru dengan nilai new\_data. |
| 2 | 12-16 | struct Node \*new\_node = new Node();  new\_node->data = new\_data; if((\*head\_ref) == NULL){  (\*head\_ref) = new\_node;  } | Mengalokasikan memori untuk sebuah node baru, menginisialisasi elemen data dengan new\_data, dan jika linked list masih kosong (ditunjukkan oleh \*head\_ref yang NULL), menjadikan node baru tersebut sebagai kepala dari linked list. |
| 2 | 19 - 24 | new\_node->next = (\*head\_ref);  new\_node->prev = NULL;  // 4. ubah prev dari head menjadi new\_node  (\*head\_ref)->prev = new\_node;  // 5. pindahkan posisi head ke new\_node  (\*head\_ref) = new\_node; | Menetapkan pointer next dari node baru ke kepala lama, mengatur prev dari kepala lama ke node baru, dan memperbarui head\_ref agar menunjuk ke node baru, menjadikannya sebagai kepala baru dari linked list. |
| 2 | 39 - 47 | // 5. Ubah next dari prev\_node sebagai new\_node  prev\_node->next = new\_node;  // 6. Ubah prev dari new\_node sebagai prev\_node  new\_node->prev = prev\_node;  // 7. Ubah prev dari node next dari new\_node  if(new\_node->next != NULL){  new\_node->next->prev = new\_node;  } | Menyesuaikan pointer next dari node sebelumnya (prev\_node) untuk menunjuk ke node baru (new\_node), mengatur pointer prev dari node baru (new\_node) untuk menunjuk ke prev\_node, dan jika node baru memiliki node setelahnya (next tidak NULL), mengatur pointer prev dari node setelahnya untuk menunjuk kembali ke new\_node. |
| 2 | 51 - 61 | 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;  } | Melakukan traversal dari awal ke akhir dalam double linked list dengan mencetak nilai data dari setiap node, kemudian melakukan traversal dari akhir ke awal dengan mencetak nilai data dari node terakhir hingga ke node pertama. |
| 3 | 4 - 8 | struct Node{  int data;  struct Node \*next;  struct Node \*prev;  }; struct Node \*new\_node = new Node();  // 2. Memasukkan data  new\_node->data = new\_data; | Mendefinisikan struktur Node untuk double linked list, mengalokasikan memori untuk sebuah node baru, dan menginisialisasi elemen data dari node baru dengan nilai new\_data. |
| 3 | 11 - 23 | Node \*new\_node = new Node();  //2. Masukkan data  new\_node->data = new\_data;  //3. Buat next dari new node sebagai head dan prev sebagai NULL  new\_node->next = (\*head\_ref);  new\_node->prev = NULL;  //4. Ubah prev dari head node menjadi new\_node  if((\*head\_ref) != NULL){  (\*head\_ref)->prev = new\_node;  }  //5. Ubah posisi head ke new\_node  (\*head\_ref) = new\_node; | Membuat sebuah node baru dengan alokasi memori, menginisialisasi data node baru dengan nilai baru, menetapkan pointer next dari node baru ke kepala lama, dan mengatur pointer prev dari kepala lama jika tidak kosong, kemudian memperbarui kepala untuk menunjuk ke node baru tersebut. |
| 3 | 28 - 33 | struct Node \*new\_node = (struct Node\*)malloc(sizeof(struct Node));  struct Node \*last = \*head\_ref; // digunakan pada step 5  //2. masukkan data  new\_node->data = new\_data;  //3. new\_node ini akan menjadi node terakhir, jadi ubah next nya ke NULL  new\_node->next = NULL; | Mengalokasikan memori untuk sebuah node baru, menginisialisasi data dari node baru dengan nilai baru, dan mengatur pointer next dari node baru ke NULL karena node tersebut akan menjadi node terakhir dalam linked list. |
| 3 | 41 - 47 | while(last->next != NULL){  last = last->next;  }  //6. ubah next dari last node  last->next = new\_node;  //7. ubah prev dari new\_node menjadi last node  new\_node->prev = last; | Mencari node terakhir dalam linked list dengan traversal hingga next dari node terakhir adalah NULL, kemudian mengatur next dari node terakhir untuk menunjuk ke node baru, dan mengatur prev dari node baru untuk menunjuk ke node terakhir. |
| 3 | 52 - 63 | 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;  } | Melakukan traversal dari awal ke akhir dalam linked list dengan mencetak nilai data dari setiap node, dan kemudian melakukan traversal dari akhir ke awal dengan mencetak nilai data dari setiap node, yang ditunjukkan oleh pointer last yang mundur melalui prev dari setiap node. |
| 4 | 4 - 8 | struct Node{  int data;  struct Node \*next;  struct Node \*prev;  }; | Struct Node untuk membuat double linked list terdiri dari elemen int data untuk menyimpan nilai, dan dua pointer struct Node \*next serta struct Node \*prev untuk menunjuk ke node berikutnya dan sebelumnya dalam list. |
| 4 | 11 - 18 | 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; | Membuat node baru dengan alokasi memori, menginisialisasi data dan pointer next dari node tersebut, serta mengatur pointer prev dari node sebelumnya jika tidak kosong, kemudian memperbarui kepala linked list untuk menunjuk ke node baru tersebut. |
| 4 | 29 - 37 | 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; | Membuat sebuah node baru dengan alokasi memori, menginisialisasi data dari node baru dengan nilai baru, dan menyisipkan node baru di antara next\_node dan next\_node->prev dalam linked list. |
| 4 | 39 - 44 | if(new\_node->prev != NULL){  new\_node->prev->next = new\_node;  } //8. jika prev new\_node adalah NULL maka new\_node akan menjadi head  else{  (\*head\_ref) = new\_node; | Mengatur pointer next dari node sebelumnya (new\_node->prev->next) untuk menunjuk ke node baru jika prev dari node baru tidak kosong, jika tidak, maka node baru menjadi kepala baru dari linked list dengan memperbarui head\_ref. |
| 4 | 48 - 59 | 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;  } | Melakukan traversal dari awal ke akhir dalam linked list dengan mencetak nilai data dari setiap node, kemudian melakukan traversal dari akhir ke awal dengan mencetak nilai data dari setiap node, yang ditunjukkan oleh pointer last yang mundur melalui prev dari setiap node. |