Jobsheet 4

Praktikum Struktur Data



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

Kode Kelas : 202323430158

Disusun Oleh :

Hanna Fadilah  
23343068

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;  }; | Mendefinisikan struktur Node untuk doubly linked list dalam bahasa C. Setiap simpul (Node) memiliki data integer (data), serta pointer next untuk simpul berikutnya dan prev untuk simpul sebelumnya dalam linked list. Struktur ini memungkinkan traversal efisien ke depan dan ke belakang dalam linked list. |
| 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;  } | Fungsi push ini efektif untuk menambahkan simpul baru ke awal dari doubly linked list. Ia mengalokasikan memori untuk simpul baru, mengatur data dan pointer next, serta mengelola pointer prev untuk memastikan keterhubungan yang tepat antara simpul-simpul dalam linked list. |
| 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;  } | Program menginisialisasi list kosong, kemudian menggunakan fungsi push untuk menambahkan nilai 6, 5, dan 2 di bagian depan list. Setelah itu, program mencetak seluruh elemen dalam list menggunakan fungsi printList. Akhirnya, program menunggu input pengguna sebelum keluar. Secara keseluruhan, kode ini menunjukkan cara membuat, menambahkan elemen, dan mencetak isi Double Linked List. |
| 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 - 12 | struct Node  {  int data;  struct Node \*next;  struct Node \*prev;  }; | Struktur Node didefinisikan untuk mewakili elemen dalam list, yang memiliki data, pointer ke node berikutnya (next), dan pointer ke node sebelumnya (prev). Program menginisialisasi list kosong, menambahkan nilai 6, 5, dan 2 di depan list menggunakan fungsi push, mencetak isi list dengan fungsi printList, dan menunggu input pengguna sebelum keluar. Kode ini menunjukkan cara membuat, menambahkan elemen, dan mencetak isi DLL. |
| 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;  } | Fungsi insertAfter memeriksa apakah node sebelumnya (prev\_node) adalah NULL. Jika ya, fungsi menampilkan pesan kesalahan dan keluar. Jika tidak, fungsi membuat node baru, mengalokasikan memori untuknya, dan menetapkan data baru. Node baru tersebut dihubungkan ke node berikutnya dari prev\_node dan mengatur prev\_node untuk menunjuk ke node baru. Jika node berikutnya dari node baru tidak NULL, maka pointer prev dari node tersebut diatur untuk menunjuk ke node baru. Kode ini menambahkan elemen ke dalam DLL setelah node yang ditentukan. |
| 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;  }  } | Fungsi printList pertama-tama mencetak data dari setiap node mulai dari kepala hingga node terakhir dalam list, lalu mencetak data dari node terakhir kembali ke kepala. Ini dilakukan dengan traversal ke depan menggunakan pointer next dan traversal ke belakang menggunakan pointer prev. Kode ini menunjukkan cara mencetak isi DLL dalam kedua arah. |
| 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;  } | Sama seperti 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. **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;  } | Fungsi push membuat node baru, mengisi data baru, dan mengatur pointer next dari node baru ke kepala list saat ini. Pointer prev dari node baru diatur ke NULL. Jika kepala list saat ini tidak NULL, pointer prev dari node tersebut diatur ke node baru. Akhirnya, kepala list diperbarui untuk menunjuk ke 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. **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;  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 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. |