Praktikum Struktur Data
Informatika | Universitas Negeri Padang
TIM DOSEN ©2023

JOBSHEET 04

Doubly Linked List

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Informatika	INF1.62.2014	2 x 50 Menit

TUJUAN PRAKTIKUM

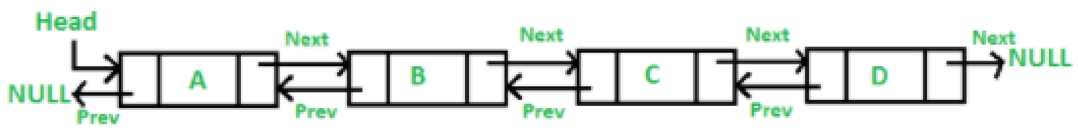
1. Mahasiswa mampu menjelaskan konsep doubly linked list
2. Mahasiswa mampu mengimplementasikan penggunaan penyimpanan data menggunakan doubly linked list

HARDWARE & SOFTWARE

1. Personal Computer
2. DevC++ IDE

TEORI SINGKAT

Doubly Linked List (DLL) berisi pointer tambahan, biasanya disebut pointer sebelumnya, bersama dengan pointer berikutnya dan data yang ada dalam daftar tertaut tunggal



Sama halnya dengan single linked list sebuah dobly link list memiliki **head** dan **tail,** perbedaannya doubly link list memiliki 2 arah data yani next dan previous.

Contoh simpul dobly linked List :

```
/* Node of a doubly linked list */
struct Node {
    int data;
    struct Node *next; // Pointer to next node in DLL
    struct Node *prev; // Pointer to previous node in DLL
};
```

Kelebihan Doubly link list dibandingkan dengan Single link list

1. DLL bisa berjalan dalam 2 arah ke depan dan kebelakang
2. Operasi penghapusan atau deletion() lebih efisien dan simple menggunakan pointer yang menunjuk ke simpul yang akan dihapus
3. Dapat melakukan proses insert() simpul lebih efisien

Dalam single link list penghapusan simpul, pointer simpul data sebelumnya diperlukan dan untuk mendapatkannya terkadang list data diulang dari awal. Berbeda dengan doubly link list kita langsung dapat pointer simpul data sebelumnya menggunakan pointer previous.

Kelemahan Dobly link list

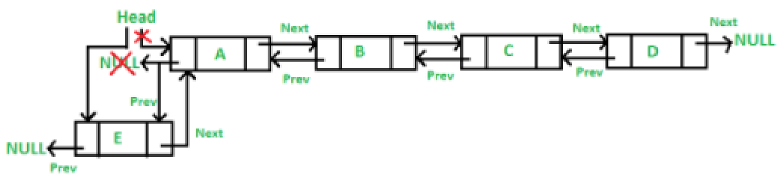
1. Setiap simpul DLL membutuhkan ruang memory tambahan untuk pointer 2 pointer
2. Setiap operasi yang dilakukan harus mencantumkan pointer simpul sebelumnya.

Jenis proses Insertion()

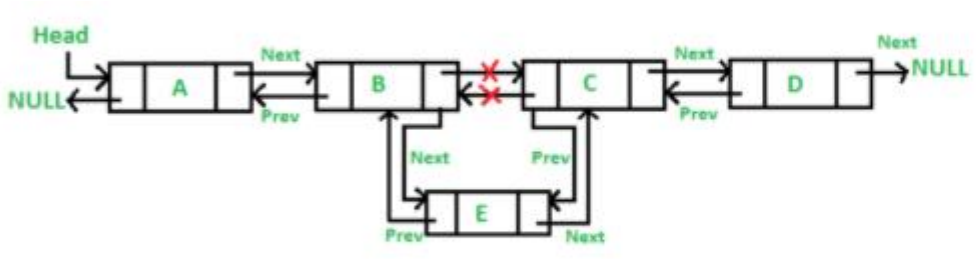
1. Penambahan simpul di depan
2. Penambahan setelah simpul yang ditentukan
3. Di simpul yang paling akhir
4. Penambahan sebelum simpul yang ditentukan

Simulasi penambahan simpul

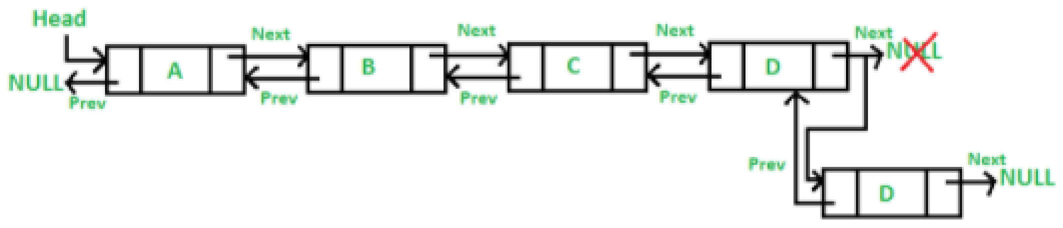
1. Penambahan di depan



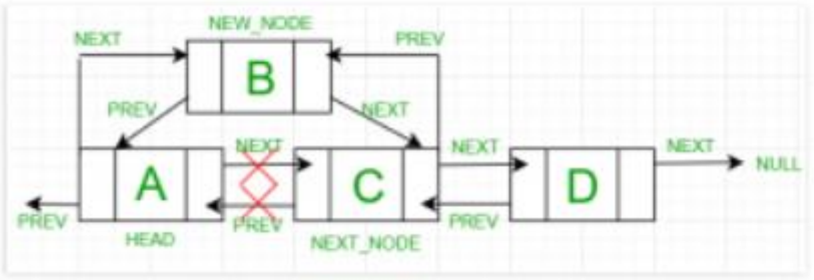
2. Penambahan simpul setelah simpul yang ditentukan



3. Penambahan simpul di akhir



4. Penambhan simpul di sebelum simpul ditentukan



PERCOBAAN

1. Insertion at front

```
#include <stdio.h>
#include <stdlib.h>

// Structure of the node
struct Node
{
    int data;
    struct Node *next; // Pointer to next node
    struct Node *prev; // Pointer to previous node
};

void push(struct Node** head_ref, int new_data)
{
    /* 1. allocate node */
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    /* 2. put in the data  */
    new_node->data = new_data;
    /* 3. Make next of new node as head and previous as NULL */
    new_node->next = (*head_ref);
    new_node->prev = NULL;
    /* 4. change prev of head node to new node */
    if ((*head_ref) != NULL)
        (*head_ref)->prev = new_node;
    /* 5. move the head to point to the new node */
    (*head_ref) = new_node;
}
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;
    }
}
int main()
{
    /* Start with the empty list */
    struct Node* head = NULL;
    push(&head, 6);
    push(&head, 5);
    push(&head, 2);
    printf("Created DLL is: ");
    printList(head);
    getchar();
    return 0;
}
```

2. Insertion After given Node

```
#include <stdio.h>
#include <stdlib.h>

// Structure of the node
struct Node
{
    int data;
    struct Node *next; // Pointer to next node
    struct Node *prev; // Pointer to previous node
```

```
};
void push(Node** head_ref, int new_data)
{
    /* 1. allocate node */
    Node* new_node = new Node();

    /* 2. put in the data */
    new_node->data = new_data;

    /* 3. Make next of new node as head and previous as NULL */
    new_node->next = (*head_ref);
    new_node->prev = NULL;

    /* 4. change prev of head node to new node */
    if ((*head_ref) != NULL)
        (*head_ref)->prev = new_node;

    /* 5. move the head to point to the new node */
    (*head_ref) = new_node;
}
void insertAfter(struct Node* prev_node, int new_data)
{
    /*1. check if the given prev_node is NULL */
    if (prev_node == NULL) {
        printf("the given previous node cannot be NULL");
        return;
    }
  /* 2. allocate new node */
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    /* 3. put in the data  */
    new_node->data = new_data;
    /* 4. Make next of new node as next of prev_node */
    new_node->next = prev_node->next;
    /* 5. Make the next of prev_node as new_node */
    prev_node->next = new_node;
    /* 6. Make prev_node as previous of new_node */
    new_node->prev = prev_node;
    /* 7. Change previous of new_node's next node */
    if (new_node->next != NULL)
        new_node->next->prev = new_node;
}
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;
    }
}
int main()
{
    /* Start with the empty list */
    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;
}
```

3. Insertion at End

```
#include <stdio.h>
#include <stdlib.h>

// Structure of the node
struct Node
{
    int data;
    struct Node *next; // Pointer to next node
    struct Node *prev; // Pointer to previous node
};
void push(Node** head_ref, int new_data)
{
    /* 1. allocate node */
    Node* new_node = new Node();
   /* 2. put in the data */
    new_node->data = new_data;
    /* 3. Make next of new node as head and previous as NULL */
    new_node->next = (*head_ref);
    new_node->prev = NULL;
    /* 4. change prev of head node to new node */
    if ((*head_ref) != NULL)
        (*head_ref)->prev = new_node;
    /* 5. move the head to point to the new node */
    (*head_ref) = new_node;
}
void append(struct Node** head_ref, int new_data)
```

```
{
    /* 1. allocate node */
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    struct Node* last = *head_ref; /* used in step 5*/
    /* 2. put in the data  */
    new_node->data = new_data;
    /* 3. This new node is going to be the last node, so
          make next of it as NULL*/
    new_node->next = NULL;
    /* 4. If the Linked List is empty, then make the new
          node as head */
    if (*head_ref == NULL) {
        new_node->prev = NULL;
        *head_ref = new_node;
        return;
    }
    /* 5. Else traverse till the last node */
    while (last->next != NULL)
        last = last->next;
    /* 6. Change the next of last node */
    last->next = new_node;
    /* 7. Make last node as previous of new node */
    new_node->prev = last;
    return;
}
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;
    }
}
int main()
{
    /* Start with the empty list */
    struct Node* head = NULL;
    // Insert 6. So linked list becomes 6->NULL
    append(&head, 6);
       // Insert 7 at the beginning. So
    // linked list becomes 7->6->NULL
    push(&head, 7);
    // Insert 1 at the beginning. So
    // linked list becomes 1->7->6->NULL
    push(&head, 1);
       // Insert 4 at the end. So linked
    // list becomes 1->7->6->4->NULL
    append(&head, 4);
    printf("Created DLL is: ");
    printList(head);
    getchar();
    return 0;
}
```

4. Insertion before given node

```
// A linked list node
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};
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;
}
/* Given a node as next_node, insert a new node before the given node */
void insertBefore(struct Node** head_ref, struct Node* next_node, int new_data)
{
    /*1. check if the given next_node is NULL */
    if (next_node == NULL) {
        printf("the given next node cannot be NULL");
        return;
    }
    /* 2. allocate new node */
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    /* 3. put in the data */
    new_node->data = new_data;
    /* 4. Make prev of new node as prev of next_node */
    new_node->prev = next_node->prev;
    /* 5. Make the prev of next_node as new_node */
    next_node->prev = new_node;
    /* 6. Make next_node as next of new_node */
    new_node->next = next_node;
    /* 7. Change next of new_node's previous node */
    if (new_node->prev != NULL)
        new_node->prev->next = new_node;
    /* 8. If the prev of new_node is NULL, it will be
       the new head node */
    else
        (*head_ref) = new_node;
}
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;
    }
}

/* Driver program to test above functions*/
int main()
{
    /* Start with the empty list */
    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;
}
```

TUGAS

1. Analisis dan buatlah penjelasan 4 program diatas dengan format berikut. File dikumpulkan dalam format .PDF

   (Catatan: Tidak perlu menjelaskan setiap baris program, cukup hal utama dari pemahaman anda terhadap program atau kumpulan dari baris program). **Berilah minimal 5 penjelasan untuk masing-masing program (total 20).**

   Perhatikan contoh dibawah!

Nomor Program	Baris Program	Petikan Source Code	Penjelasan

1	5 – 10	`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

2			

DAFTAR PUSTAKA

1. Kernighan, Brian W, & Ritchie, Dennis M. 1988. The Ansi C Programming Language Second Edition, Prentice-Hall.
2. Cipta Ramadhani. 2015. Dasar Algoritma & Struktur Data. Yogyakarta: ANDI.