Linked List

Linked list adalah salah satu struktur data yang digunakan untuk menyimpan dan mengelola data secara dinamis. Berbeda dengan array yang menggunakan alokasi memori statis, linked list menggunakan alokasi memori dinamis dan terdiri dari simpul-simpul yang saling terhubung. Dalam bahasa C, linked list dapat diimplementasikan dengan menggunakan struktur (struct) untuk merepresentasikan simpul-simpulnya. Setiap simpul dalam linked list menyimpan dua informasi utama: data dan referensi atau pointer ke simpul berikutnya.

Berikut adalah contoh struktur untuk merepresentasikan simpul dalam linked list:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>S

// Struktur untuk merepresentasikan skill
struct listskill {
    char name[50];
    int damage;
    int mana;
    struct listskill* next;
};
```

Struktur data dasar dalam linked list disebut node dalam code tersebut nodenya adalah listskill. Setiap node terdiri dari dua bagian: data dan alamat (pointer) ke node berikutnya. Nama, damage, mana adalah informasi atau nilai yang ingin disimpan dalam simpul, dan next adalah pointer yang menunjuk ke simpul berikutnya dalam linked list.

```c
// Fungsi untuk membuat node baru (skill dalam listskill)
struct listskill* createskill(const char* name, int damage, int mana) {
    struct listskill* newItem = (struct listskill*)malloc(sizeof(struct listskill));
    if (newItem == NULL) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }
    strcpy(newItem->name, name);
    newItem->damage = damage;
    newItem->mana = mana;
    newItem->next = NULL;
    return newItem;
}
```

Fungsi ini dapat digunakan untuk membuat node baru dalam linked list skill. Dengan memanggil fungsi ini dapat membuat node baru dengan memberikan nilai untuk name, damage, dan mana.

```c
// Fungsi untuk menambahkan skill ke listskill (linked list)
void addskill(struct listskill** head, const char* name, int damage, int mana) {
    struct listskill* newItem = createskill(name, damage, mana);
    newItem->next = *head;
    *head = newItem;
}
```

Fungsi ini dideklarasikan dengan tipe pengembalian void yang berarti tidak mengembalikan nilai. Parameter pertamanya adalah pointer ke pointer ke struct listskill, yang merujuk ke pointer yang menunjuk ke kepala dari linked list. Parameter selanjutnya adalah data yang akan dimasukkan ke linked list. Fungsi createskill (yang tidak diberikan dalam potongan code ini) digunakan untuk membuat simpul baru dengan data yang diberikan (nama, damage, dan mana). Hasil dari fungsi ini adalah pointer ke simpul baru yang dibuat. Setelah simpul baru dibuat, pointer next dari simpul tersebut diatur untuk menunjuk ke simpul yang sebelumnya menjadi kepala linked list. Kemudian, pointer kepala linked list (*head) diubah untuk menunjuk ke simpul baru. Dengan cara ini, simpul baru telah ditambahkan di awal linked list. Saat fungsi ini dipanggil, elemen baru akan selalu ditambahkan di awal linked list. Jadi, elemen yang ditambahkan terakhir akan menjadi elemen pertama yang diakses saat traversing linked list.

```c
// Fungsi untuk menampilkan isi skill (linked list)
void displayskill(struct listskill* head) {
    struct listskill* current = head;
    printf("Skill:\n");
    while (current != NULL) {
        printf("%-20s damage: %-5d mana: %-7.2d\n", current->name, current->damage, current->mana);
        current = current->next;
    }
    printf("\n");
}
```

fungsi displayskill yang bertujuan untuk menampilkan isi dari linked list skill (listskill). Fungsi ini melakukan iterasi melalui linked list dan mencetak informasi setiap node ke layar. Fungsi ini dapat digunakan untuk menampilkan isi linked list skill. Misalnya, jika head adalah pointer ke node pertama dalam linked list, maka pemanggilan fungsi displayskill(head) akan mencetak semua informasi skill dalam linked list tersebut ke layar.

```c
int main() {
    // Inisialisasi listskill (linked list) kosong
    struct listskill* skill = NULL;

    // Menambahkan beberapa skill ke dalam listskill
    addskill(&skill, "bomb", 50, 20);
    addskill(&skill, "slash", 30, 20);
    addskill(&skill, "serious punch", 99, 70);

    // Menampilkan isi listskill
    displayskill(skill);

    return 0;
}
```

Code di atas adalah implementasi main program dalam bahasa C yang menggunakan linked list untuk merepresentasikan sebuah listskill. Dimulai dengan mendeklarasikan dan

menginisialisasi pointer skill sebagai linked list. Pada awalnya, linked list ini kosong, artinya tidak memiliki elemen. Selanjutnya, tiga item (skill) ditambahkan ke linked list menggunakan fungsi addskill. Setiap pemanggilan addskill menambahkan skill baru ke awal linked list. Terakhir, fungsi displayskill dipanggil untuk menampilkan isi linked list ke layar. Fungsi ini melakukan iterasi melalui linked list dan menampilkan informasi dari setiap elemen (skill) seperti nama, damage, dan mana.

Queue

Queue adalah struktur data yang mengikuti konsep "First In, First Out" (FIFO), artinya elemen pertama yang dimasukkan ke dalam queue akan menjadi elemen pertama yang keluar dari queue. Implementasi dasar dari queue dapat menggunakan array atau linked list.

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

// Struktur untuk merepresentasikan elemen dalam queue
struct QueueNode {
    int data;
    struct QueueNode* next;
};

// Struktur untuk merepresentasikan queue itu sendiri
struct Queue {
    struct QueueNode* front; // Pointer ke elemen depan queue
    struct QueueNode* rear;  // Pointer ke elemen belakang queue
};

// Fungsi untuk membuat node baru dalam queue
struct QueueNode* createQueueNode(int data) {
    struct QueueNode* newNode = (struct QueueNode*)malloc(sizeof(struct QueueNode));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Fungsi untuk membuat queue baru
struct Queue* createQueue() {
    struct Queue* queue = (struct Queue*)malloc(sizeof(struct Queue));
    queue->front = queue->rear = NULL;
    return queue;
}
```

Struktur QueueNode dan Queue. struct QueueNode digunakan untuk merepresentasikan setiap elemen dalam queue, dan struct Queue digunakan untuk merepresentasikan queue itu sendiri. Struktur Queue memiliki dua pointer, yaitu front yang menunjuk ke elemen depan dan rear yang menunjuk ke elemen belakang queue. Fungsi createQueueNode digunakan untuk membuat node baru dalam queue dengan nilai data tertentu sedangkan Fungsi createQueue digunakan untuk membuat queue baru dengan pointer front dan rear diatur sebagai NULL.

```c
// Fungsi untuk menambahkan elemen ke dalam queue
void enqueue(struct Queue* queue, int data) {
    struct QueueNode* newNode = createQueueNode(data);

    // Jika queue kosong, elemen baru menjadi elemen pertama
    if (queue->rear == NULL) {
        queue->front = queue->rear = newNode;
        return;
    }

    // Menambahkan elemen baru ke belakang queue
    queue->rear->next = newNode;
    queue->rear = newNode;
}
```

Fungsi enqueue menambahkan elemen baru ke dalam queue. Jika queue kosong, elemen baru menjadi elemen pertama. Jika tidak, elemen baru ditambahkan ke belakang queue

```c
// Fungsi untuk menghapus elemen dari depan queue
void dequeue(struct Queue* queue) {
    // Jika queue kosong, tidak ada yang dihapus
    if (queue->front == NULL)
        return;

    struct QueueNode* temp = queue->front;

    // Memindahkan pointer front ke elemen berikutnya
    queue->front = queue->front->next;

    // Jika front menjadi NULL, artinya queue kosong, rear juga diatur NULL
    if (queue->front == NULL)
        queue->rear = NULL;

    // Membebaskan memori dari elemen yang dihapus
    free(temp);
}
```

Fungsi dequeue menghapus elemen dari depan queue. Jika queue tidak kosong, pointer front dipindahkan ke elemen berikutnya. Jika setelah penghapusan front menjadi NULL, artinya queue kosong, maka pointer rear juga diatur NULL.

```c
// Fungsi untuk menampilkan isi dari queue
void displayQueue(struct Queue* queue) {
    if (queue->front == NULL) {
        printf("Queue is empty\n");
        return;
    }

    struct QueueNode* current = queue->front;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}
```

Fungsi displayQueue menampilkan isi dari queue. Jika queue kosong, pesan "Queue is empty" akan ditampilkan. Jika tidak, fungsi ini melakukan iterasi melalui elemen-elemen queue dan menampilkan nilai datanya.

```c
// Fungsi utama (main)
int main() {
    // Membuat queue baru
    struct Queue* myQueue = createQueue();

    // Menambahkan elemen ke dalam queue
    enqueue(myQueue, 10);
    enqueue(myQueue, 20);
    enqueue(myQueue, 30);

    // Menampilkan isi queue
    printf("Queue elements: ");
    displayQueue(myQueue);

    // Menghapus elemen dari depan queue
    dequeue(myQueue);

    // Menampilkan isi queue setelah penghapusan
    printf("Queue elements after dequeue: ");
    displayQueue(myQueue);

    return 0;
}
```

Program utama menciptakan queue baru, menambahkan beberapa elemen ke dalamnya, menampilkan isi queue, menghapus elemen pertama, dan menampilkan isi queue setelah penghapusan.