LAPORAN LINKED LIST

Oleh:

Nama	: L HAFIDL ALKHAIR
NIM	: 2023903430060
Kelas	: TRKJ 1.C
Jurusan	: Teknologi Informasi dan Komputer
Program Studi	: Teknologi Rekayasa Komputer Jaringan
Dosen Pembimbing	: Indrawati, SST. MT

JURUSAN TEKNOLOGI, KOMPUTER, DAN INFORMASI

PRODI TEKNOLOGI REKAYASA KOMPUTER DAN JARINGAN

POLITEKNIK NEGERI LHOKSEUMAWE

TAHUN AJARAN 2023/2024

LEMBAR PENGESAHAN

No. Praktikum	: 06 /TIK/TRKJ-1C/ Data Structure And Algorithms Practice
Judul	: Laporan Linked List
Nama	: L HAFIDL ALKHAIR
NIM	: 2023903430060
Kelas	: TRKJ-1C
Jurusan	: Teknologi Informasi Dan Komputer
Prodi	: Teknologi Rekayasa Komputer Jaringan
Tanggal Praktikum	: 16 Oktober 2023
Tanggal Penyerahan	: 30 Oktober 2023

Buketrata, 30 Oktober 2023

Dosen Pembimbing,

Indrawati, SST.MT

Nip. 19740815 200112 2 001

A. TUJUAN

Linked list adalah struktur data yang digunakan dalam pemrograman komputer untuk menyimpan koleksi data. Tujuan dari menggunakan linked list dalam konteks mahasiswa atau data mahasiswa adalah untuk menyimpan informasi tentang mahasiswa-mahasiswa tersebut secara dinamis. Berikut adalah beberapa tujuan penggunaan linked list untuk mahasiswa:

1. **Penyimpanan Dinamis:** Linked list memungkinkan penyimpanan data mahasiswa secara dinamis, artinya kita dapat menambah atau menghapus data mahasiswa dengan mudah tanpa harus mengalokasikan ruang memori secara statis.

2. **Pengelolaan Memori:** Linked list memungkinkan penggunaan memori secara efisien karena ruang memori hanya dialokasikan ketika diperlukan. Ini membantu menghindari pemborosan memori.

3. **Penyisipan dan Penghapusan Data:** Operasi penyisipan dan penghapusan elemen dalam linked list lebih efisien dibandingkan dengan array statis. Linked list memungkinkan penyisipan atau penghapusan elemen di tengah-tengah list dengan cepat tanpa memerlukan pergeseran data.

4. **Pencarian Data:** Linked list memungkinkan pencarian data dengan cara yang efisien, terutama jika data terurut. Meskipun pencarian dalam linked list tidak secepat array terurut, tetapi dalam beberapa kasus, linked list dapat diurutkan untuk meningkatkan efisiensi pencarian.

5. **Implementasi Struktur Data Lain:** Linked list adalah dasar untuk struktur data lain seperti stack, queue, dan deque. Dengan menggunakan linked list, mahasiswa dapat memahami konsep dasar dari struktur data ini.

6. **Pengembangan Aplikasi Database:** Dalam aplikasi pengelolaan data mahasiswa atau sistem informasi akademik, linked list dapat digunakan sebagai

dasar untuk mengimplementasikan basis data sederhana, terutama jika ada banyak operasi penyisipan, penghapusan, dan pencarian yang perlu dilakukan.

7. **Pengembangan Keterampilan Pemrograman**:** Memahami konsep linked list membantu mahasiswa dalam pengembangan keterampilan pemrograman dan pemahaman mendalam tentang bagaimana struktur data bekerja di tingkat dasar.

Jadi, menggunakan linked list dalam konteks data mahasiswa membantu dalam pengelolaan dan manipulasi data dengan cara yang efisien dan fleksibel.

B. DASAR TEORI

Secara teori, linked list adalah sejumlah node yang dihubungkan secara linier dengan bantuan pointer. Dikatakan singly (single) linked apabila hanya ada satu pointer yang menghubungkan setiap node. Singly artinya field pointer-nya hanya satu buah saja dan satu arah.

Senarai berkait adalah struktur data yang paling dasar. Senarai berkait terdiri atas sejumlah unsur-unsur dikelompokkan, atau terhubung, bersama-sama di suatu deret yang spesifik. Senarai berkait bermanfaat di dalam memelihara koleksi-koleksi data, yang serupa dengan array/larik yang sering digunakan. Bagaimanapun juga, senarai berkait memberikan keuntungan-keuntungan penting yang melebihi array/larik dalam banyak hal. Secara rinci, senarai berkait lebih efisien di dalam melaksanakan penyisipan-penyisipan dan penghapusan-penghapusan. Senarai berkait juga menggunakan alokasi penyimpanan secara dinamis, yang merupakan penyimpanan yang dialokasikan pada runtime. Karena di dalam banyak aplikasi, ukuran dari data itu tidak diketahui pada saat kompile, hal ini bisa merupakan suatu atribut yang baik juga.

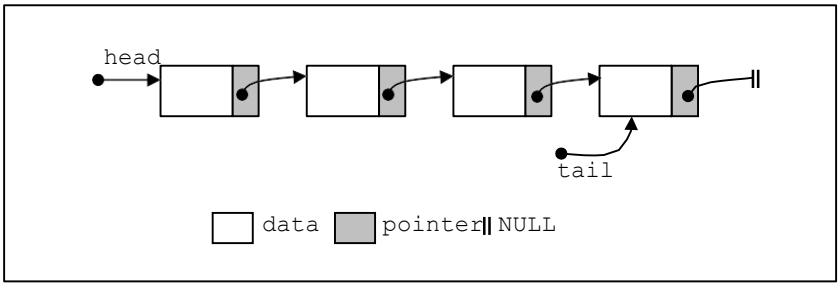
Linked List atau juga biasa dalam Bahasa Indonesia disebut "Senarai Berantai" adalah struktur data yang terdiri dari urutan record data dimana setiap record memiliki field yang menyimpan alamat atau referensi dari record selanjutnya sesuai

dengan urutan. Node merupakan suatu sebutan dari elemen data yang dihubungkan dengan link pada linked list. Biasanya linked list menggunakan pointer.

Dalam pembelajaran struktur data, kita akan lebih sering mengenal dengan istilah:

1. Push untuk menambah data.
    - PushHead – Menambah data ke barisan paling awal
    - PushTail – Menambah data ke barisan paling akhir
    - PushMid – Menambah data ke barisan di tengah (sorting)
2. Pop untuk menghapus data.
    - PopHead – Menghapus data paling awal
    - PopTail – Menghapus data paling akhir
    - PopMid – Menghapus data ditengah (sesuai parameter value)

Single linked list atau biasa disebut linked list terdiri dari elemen-elemen individu, dimana masing-masing dihubungkan dengan pointer tunggal. Masing-masing elemen terdiri dari dua bagian, yaitu sebuah data dan sebuah pointer yang disebut dengan pointer next. Dengan menggunakan struktur two-member seperti ini, linked list dibentuk dengan cara menunjuk pointer next suatu elemen ke elemen yang mengikutinya seperti gambar 1.5. Pointer next pada elemen terakhir merupakan NULL, yang menunjukkan akhir dari suatu list. Elemen pada awal suatu list disebut head, dan elemen terakhir dari suatu list disebut tail.



C. ALAT DAN BAHAN

Di dalam pembuatan sebuah Bahasa pemrograman tentu saja memerlukan alat dan bahan-bahan supaya program tersebut dapat di jalankan dengan baik dan tidak mengalami yang namanya error saat program atau kode tersebut di jalankan. Adapun alat dan bahan sebagai berikut:

1. Laptop atau device

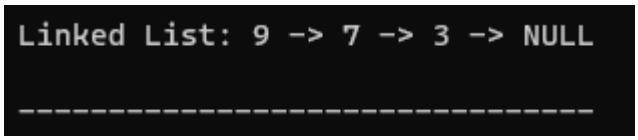
2. Tools atau aplikasi pemrograman (disini kita memakai aplikasi Dev-C++)

D. PROGRAM ARRAY

1. Menambahkan Elemen ke Awal Single Linked List.

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

// Struktur untuk merepresentasikan node dalam linked list
struct Node {
    int data;
    struct Node* next;
};

// Fungsi untuk menambahkan elemen di awal linked list
void insertAtBeginning(struct Node** head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}

// Fungsi untuk menampilkan elemen-elemen dalam linked list
void display(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;

    // Menambahkan elemen ke awal linked list
    insertAtBeginning(&head, 3);
    insertAtBeginning(&head, 7);
    insertAtBeginning(&head, 9);

    // Menampilkan linked list
    printf("Linked List: ");
    display(head);

    return 0;
}
```

Hasil Ouput



Analisa :

Program mendefinisikan struktur data Node yang memiliki dua anggota: data untuk menyimpan nilai dan next yang merupakan pointer ke node berikutnya dalam linked list.

- Fungsi insertAtBeginning: digunakan untuk menambahkan elemen baru ke awal linked list.
- Menerima dua parameter: alamat dari pointer yang menunjuk ke head linked list (head) dan nilai data yang ingin dimasukkan (data).
- Fungsi membuat node baru, mengatur nilai data, dan menetapkan next untuk menunjuk ke node yang saat ini menjadi head.
- Pointer head diperbarui untuk menunjuk ke node baru, sehingga node baru menjadi elemen pertama dalam linked list.
- Fungsi Display digunakan untuk menampilkan elemen-elemen dalam linked list.
- Menerima parameter head yang merupakan pointer ke node pertama dalam linked list.
- Fungsi menggunakan loop while untuk mencetak nilai setiap node hingga mencapai node terakhir (node terakhir memiliki next yang menunjuk ke NULL).
- Fungsi main adalah titik masuk utama program.
- Dalamnya, variabel pointer head digunakan untuk menyimpan alamat node pertama dalam linked list.
- Program menambahkan tiga elemen ke awal linked list menggunakan fungsi insertAtBeginning.
- Setelah itu, program menampilkan linked list menggunakan fungsi display.

2. Menghapus Elemen dari single Linked List

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

// Struktur untuk merepresentasikan node dalam linked list
struct Node {
    int data;
    struct Node* next;
};

// Fungsi untuk menambahkan elemen di awal linked list
void insertAtBeginning(struct Node** head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}

// Fungsi untuk menampilkan elemen-elemen dalam linked list
void display(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

// Fungsi untuk menghapus elemen dari linked list berdasarkan nilai
void deleteNode(struct Node** head, int key) {
    struct Node *current = *head, *prev = NULL;

    // Jika node pertama memiliki nilai yang sesuai
    if (current != NULL && current->data == key) {
        *head = current->next;
        free(current);
        printf("Node with value %d deleted from the linked list.\n", key);
        return;
    }

    // Mencari node dengan nilai yang sesuai
    while (current != NULL && current->data != key) {
        prev = current;
        current = current->next;
    }

    // Jika node tidak ditemukan
    if (current == NULL) {
        printf("Node with value %d not found in the linked list.\n", key);
        return;
    }

    // Menghapus node yang ditemukan
    prev->next = current->next;
```

```c
    free(current);
    printf("Node with value %d deleted from the linked list.\n", key);
}

int main() {
    struct Node* head = NULL;

    // Menambahkan elemen ke awal linked list
    insertAtBeginning(&head, 3);
    insertAtBeginning(&head, 7);
    insertAtBeginning(&head, 9);

    // Menampilkan linked list
    printf("Linked List: ");
    display(head);

    // Menghapus node dengan nilai 7
    deleteNode(&head, 7);

    // Menampilkan linked list setelah penghapusan
    printf("Linked List after deletion: ");
    display(head);

    // Mengosongkan memori yang digunakan oleh linked list (tidak ditampilkan dalam contoh ini)
    return 0;
}
```

Hasil Ouput

```
Linked List: 9 -> 7 -> 3 -> NULL
Node with value 7 deleted from the linked list.
Linked List after deletion: 9 -> 3 -> NULL

--------------------------------
```

Analisa ;

Program mendefinisikan struktur data Node yang memiliki dua anggota: data untuk menyimpan nilai, dan next yang merupakan pointer ke node berikutnya dalam linked list.

- Fungsi insertAtBeginning: digunakan untuk menambahkan elemen baru ke awal linked list.
- Menerima dua parameter: alamat dari pointer yang menunjuk ke head linked list (head) dan nilai data yang ingin dimasukkan (data).
- Fungsi membuat node baru, mengatur nilai data, dan menetapkan next untuk menunjuk ke node yang saat ini menjadi head.

- Pointer head diperbarui untuk menunjuk ke node baru, sehingga node baru ini sekarang menjadi elemen pertama dalam linked list.
- Fungsi display: digunakan untuk menampilkan elemen-elemen dalam linked list.
- Menerima parameter head yang merupakan pointer ke node pertama dalam linked list.
- Fungsi ini menggunakan loop while untuk mencetak nilai setiap node hingga mencapai node terakhir (node terakhir memiliki next yang menunjuk ke NULL).
- Fungsi deleteNode: digunakan untuk menghapus node dengan nilai tertentu dari linked list.
- Menerima dua parameter: alamat dari pointer yang menunjuk ke head linked list (head) dan nilai yang ingin dihapus (key).
- Fungsi ini mencari node dengan nilai yang sesuai, mengubah pointer next dari node sebelumnya untuk menghapus node tersebut, dan kemudian membebaskan memori yang dialokasikan untuk node yang dihapus.
- Fungsi main adalah titik masuk utama program.
- Dalamnya, variabel pointer head digunakan untuk menyimpan alamat node pertama dalam linked list.
- Program menambahkan tiga elemen ke awal linked list menggunakan fungsi insertAtBeginning, kemudian menghapus node dengan nilai 7 menggunakan fungsi deleteNode.
- Program menampilkan isi linked list sebelum dan sesudah penghapusan.

3. Mencari Elemen dalam Single Linked List:

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

// Struktur untuk merepresentasikan node dalam linked list
struct Node {
    int data;
    struct Node* next;
};

// Fungsi untuk menambahkan elemen di awal linked list
void insertAtBeginning(struct Node** head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}

// Fungsi untuk menampilkan elemen-elemen dalam linked list
void display(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
// Fungsi untuk mencari elemen dalam linked list
int search(struct Node* head, int key) {
    struct Node* current = head;
    int position = 1;

    // Mencari elemen dengan nilai yang sesuai
    while (current != NULL) {
        if (current->data == key) {
            printf("Node with value %d found at position %d.\n", key, position);
            return 1; // Nilai ditemukan
        }
        current = current->next;
        position++;
    }

    // Jika nilai tidak ditemukan
    printf("Node with value %d not found in the linked list.\n", key);
    return 0; // Nilai tidak ditemukan
}

int main() {
    struct Node* head = NULL;

    // Menambahkan elemen ke awal linked list
    insertAtBeginning(&head, 3);
    insertAtBeginning(&head, 7);
    insertAtBeginning(&head, 9);
```

```
55
56      // Menampilkan linked list
57      printf("Linked List: ");
58      display(head);
59
60      // Mencari nilai 7 dalam linked list
61      int key = 7;
62      if (search(head, key)) {
63          printf("Element %d ditemukan di linked list.\n", key);
64      } else {
65          printf("Element %d tidak ditemukan di linked list.\n", key);
66      }
67
68      return 0;
69  }
```

Hasil Output

```
Linked List: 9 -> 7 -> 3 -> NULL
Node with value 7 found at position 2.
Element 7 ditemukan di linked list.

--------------------------------
```

Analisa :

Program mendefinisikan struktur data Node yang memiliki dua anggota: data untuk menyimpan nilai, dan next yang merupakan pointer ke node berikutnya dalam linked list.

- Fungsi insertAtBeginning: digunakan untuk menambahkan elemen baru ke awal linked list.
- Menerima dua parameter: alamat dari pointer yang menunjuk ke head linked list (head) dan nilai data yang ingin dimasukkan (data).
- Fungsi membuat node baru, mengatur nilai data, dan menetapkan next untuk menunjuk ke node yang saat ini menjadi head.
- Pointer head diperbarui untuk menunjuk ke node baru, sehingga node baru ini sekarang menjadi elemen pertama dalam linked list.

- Fungsi display: digunakan untuk menampilkan elemen-elemen dalam linked list.
- Menerima parameter head yang merupakan pointer ke node pertama dalam linked list.
- Fungsi ini menggunakan loop while untuk mencetak nilai setiap node hingga mencapai node terakhir (node terakhir memiliki next yang menunjuk ke NULL).
- Fungsi search: digunakan untuk mencari elemen dalam linked list berdasarkan nilai (key).
- Menerima dua parameter: pointer ke head linked list (head) dan nilai yang ingin dicari (key).
- Fungsi ini menggunakan loop while untuk mencari nilai dalam linked list.
- Jika nilai ditemukan, fungsi mengembalikan 1. Jika tidak, fungsi mengembalikan 0.
- Fungsi main adalah titik masuk utama program.
- Dalamnya, variabel pointer head digunakan untuk menyimpan alamat node pertama dalam linked list.
- Program menambahkan beberapa elemen ke awal linked list menggunakan fungsi insertAtBeginning, kemudian mencari nilai 7 dalam linked list menggunakan fungsi search.
- Program menampilkan apakah nilai tersebut ditemukan atau tidak dalam linked list.

4. **Menghitung Jumlah Elemen dalam Single Linked List:**

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

// Struktur untuk merepresentasikan node dalam linked list
struct Node {
    int data;
    struct Node* next;
};

// Fungsi untuk menambahkan elemen di awal linked list
void insertAtBeginning(struct Node** head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}

// Fungsi untuk menampilkan elemen-elemen dalam linked list
void display(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

// Fungsi untuk menghitung jumlah elemen dalam linked list
int countNodes(struct Node* head) {
    int count = 0;
    struct Node* current = head;

    // Menghitung jumlah node dalam linked list
    while (current != NULL) {
        count++;
        current = current->next;
    }

    return count;
}
```

```
41
42  int main() {
43      struct Node* head = NULL;
44
45      // Menambahkan elemen ke awal linked list
46      insertAtBeginning(&head, 3);
47      insertAtBeginning(&head, 7);
48      insertAtBeginning(&head, 9);
49
50      // Menampilkan linked list
51      printf("Linked List: ");
52      display(head);
53
54      // Menghitung jumlah elemen dalam linked list
55      int nodeCount = countNodes(head);
56      printf("Jumlah Node dalam linked list: %d\n", nodeCount);
57      public int __cdecl printf (const char * __restrict__ _Format, ...)
58      return 0;
59  }
```

Hasil Output

```
Linked List: 9 -> 7 -> 3 -> NULL
Jumlah Node dalam linked list: 3

--------------------------------
```

Analisa

Program mendefinisikan struktur data Node yang memiliki dua anggota: data untuk menyimpan nilai, dan next yang merupakan pointer ke node berikutnya dalam linked list.

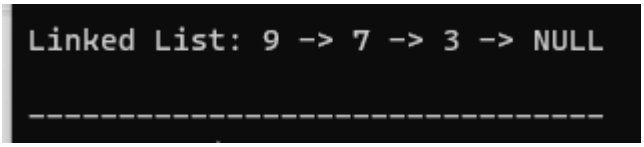
- Fungsi insertAtBeginning: digunakan untuk menambahkan elemen baru ke awal linked list.
- Menerima dua parameter: alamat dari pointer yang menunjuk ke head linked list (head) dan nilai data yang ingin dimasukkan (data).
- Fungsi membuat node baru, mengatur nilai data, dan menetapkan next untuk menunjuk ke node yang saat ini menjadi head.

- Pointer head diperbarui untuk menunjuk ke node baru, sehingga node baru ini sekarang menjadi elemen pertama dalam linked list.
- Fungsi display: digunakan untuk menampilkan elemen-elemen dalam linked list.
- Menerima parameter head yang merupakan pointer ke node pertama dalam linked list.
- Fungsi ini menggunakan loop while untuk mencetak nilai setiap node hingga mencapai node terakhir (node terakhir memiliki next yang menunjuk ke NULL).
- Fungsi countNodes: digunakan untuk menghitung jumlah elemen dalam linked list.
- Menerima parameter head yang merupakan pointer ke node pertama dalam linked list.
- Fungsi ini menggunakan loop while untuk menghitung jumlah node dalam linked list dan mengembalikan nilai tersebut.
- Fungsi main adalah titik masuk utama program.
- Dalamnya, variabel pointer head digunakan untuk menyimpan alamat node pertama dalam linked list.
- Program menambahkan beberapa elemen ke awal linked list menggunakan fungsi insertAtBeginning, kemudian menghitung jumlah elemen dalam linked list menggunakan fungsi countNodes.
- Program kemudian menampilkan jumlah elemen dalam linked list.

5. Membuat dan Menampilkan Single Linked List

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

// Struktur untuk merepresentasikan node dalam linked list
struct Node {
    int data;
    struct Node* next;
};

// Fungsi untuk menambahkan elemen di awal linked list
void insertAtBeginning(struct Node** head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}

// Fungsi untuk menampilkan elemen-elemen dalam linked list
void display(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;

    // Menambahkan elemen ke awal linked list
    insertAtBeginning(&head, 3);
    insertAtBeginning(&head, 7);
    insertAtBeginning(&head, 9);

    // Menampilkan linked list
    printf("Linked List: ");
    display(head);

    return 0;
}
```

Hasil Ouput :



Analisa :

Program mendefinisikan struktur data Node yang memiliki dua anggota: data untuk menyimpan nilai, dan next yang merupakan pointer ke node berikutnya dalam linked list.

- Fungsi insertAtBeginning: Fungsi ini digunakan untuk menambahkan elemen baru ke awal linked list.
- Menerima dua parameter: alamat dari pointer yang menunjuk ke head linked list (head) dan nilai data yang ingin dimasukkan (data).
- Fungsi membuat node baru, mengatur nilai data, dan menetapkan next untuk menunjuk ke node yang saat ini menjadi head.
- Pointer head diperbarui untuk menunjuk ke node baru, sehingga node baru ini sekarang menjadi elemen pertama dalam linked list.
- Fungsi display: digunakan untuk menampilkan elemen-elemen dalam linked list.
- Menerima parameter head yang merupakan pointer ke node pertama dalam linked list.
- Fungsi ini menggunakan loop while untuk mencetak nilai setiap node hingga mencapai node terakhir (node terakhir memiliki next yang menunjuk ke NULL).
- Fungsi main adalah titik masuk utama program.
- Dalamnya, variabel pointer head digunakan untuk menyimpan alamat node pertama dalam linked list.
- Program menambahkan beberapa elemen ke awal linked list menggunakan fungsi insertAtBeginning, kemudian menampilkan linked list menggunakan fungsi display.

PENUTUP

KESIMPULAN

Dalam pemrograman C, implementasi linked list adalah konsep yang mendasar dan penting. Linked list merupakan struktur data dinamis yang memungkinkan penyimpanan dan pengelolaan data dengan fleksibilitas. Melalui penggunaan pointer, linked list mengatasi batasan array dalam hal ukuran yang tetap. Oleh karena itu, linked list sangat berguna dalam situasi di mana jumlah dan ukuran data tidak pasti atau perlu diubah secara dinamis selama runtime program.

Pada dasarnya, linked list terdiri dari node-node yang saling terhubung melalui pointer. Node menyimpan data dan alamat (pointer) ke node berikutnya. Ada beberapa jenis linked list, termasuk single linked list, double linked list, dan circular linked list, masing-masing dengan karakteristik dan kegunaannya sendiri.

Pemahaman konsep linked list di C melibatkan pengelolaan node, alokasi dan dealokasi memori secara dinamis, serta penanganan pointer dengan hati-hati untuk mencegah kebocoran memori atau kesalahan dalam penunjukan data.

Dalam mengimplementasikan linked list, sangat penting untuk memahami bagaimana operasi dasar seperti penambahan, penghapusan, pencarian, dan penampilan data dilakukan melalui manipulasi pointer dan struktur data. Dengan memahami konsep ini, pengembang dapat memanfaatkan kelebihan linked list dalam memecahkan berbagai masalah pemrograman yang melibatkan manipulasi data yang dinamis.