Artikel Praktikum Struktur Data

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

Disusun Oleh :
Hanna Fadilah
23343068

PROGRAM STUDI INFORMATIKA (NK)
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2023

A.Pointer, struct dan array

1.Pointer

Pointer adalah variabel yang menyimpan alamat memori dari variabel lain atau lokasi memori tertentu dalam sistem komputer. Pointer memungkinkan pengguna untuk mengakses dan memanipulasi data secara langsung melalui alamat memori, yang memungkinkan pemrogram untuk melakukan operasi yang lebih kompleks dan efisien.

Sebagai contoh, ketika kita mendeklarasikan variabel bertipe int seperti berikut:

```
int angka = 10;
```

Maka, kita dapat membuat pointer untuk menyimpan alamat memori variabel tersebut:

```
int *ptrAngka = &angka;
```

Dalam contoh di atas, ptrAngka adalah pointer yang menyimpan alamat memori dari variabel `angka`.

a) Alamat Memori: Pointer adalah variabel yang menyimpan alamat memori dari variabel lain atau objek dalam memori komputer. Setiap variabel di dalam program komputer diberi alamat unik dalam memori fisik, dan pointer memungkinkan akses langsung ke alamat tersebut.
b) Manipulasi Memori: Penggunaan pointer memungkinkan manipulasi memori secara langsung. Ini termasuk alokasi dan dealokasi memori, pengalihan data antar struktur data, dan akses langsung ke data dalam memori
c) Referensi Indeks: Pointer memungkinkan referensi indirek ke variabel atau objek lain. Alih-alih menyimpan nilai variabel itu sendiri, pointer menyimpan alamat di mana nilai tersebut disimpan.

2.Struct (Struktur)

Struct, singkatan dari "structure", adalah tipe data yang memungkinkan pengguna untuk menggabungkan beberapa tipe data yang berbeda menjadi satu kesatuan logis. Dalam struktur, kita dapat mendefinisikan atribut-atribut atau variabel-variabel yang berbeda dengan tipe data yang berbeda pula. Struktur biasanya digunakan untuk merepresentasikan objek atau entitas dalam program.

a) Penyatuan Data: Struktur memungkinkan pengguna untuk mengelompokkan data terkait dalam satu unit. Ini memungkinkan untuk merepresentasikan objek dalam program dengan cara yang lebih terstruktur, mirip dengan bagaimana kita memandang objek dalam kehidupan nyata. Misalnya, jika kita ingin merepresentasikan sebuah mobil dalam program, kita dapat menggunakan struktur yang memiliki atribut seperti model, tahun, dan warna.
b) Definisi Atribut: Dalam struktur, kita mendefinisikan atribut-atribut atau variabel-variabel yang terkait dengan objek yang ingin direpresentasikan. Setiap atribut dapat memiliki tipe data yang berbeda, seperti integer, float, string, atau bahkan tipe data yang lebih kompleks seperti array atau pointer.
c) Pengaksesan Data: Setelah struktur didefinisikan, kita dapat membuat variabel dari tipe struktur tersebut dan mengakses atribut-atributnya menggunakan operator titik.

Contoh deklarasi struct :

```
struct Mahasiswa {
    char nama[50];
    int usia;
     char jurusan[50];
};
```

3.Array

Array adalah struktur data yang terdiri dari kumpulan elemen yang sama jenisnya, yang disusun secara berurutan dalam memori komputer dan dapat diakses menggunakan indeks. Setiap elemen dalam array diberi nomor indeks yang unik, dimulai dari nol, yang digunakan untuk mengidentifikasi dan mengakses elemen tertentu dalam array.

Kumpulan Elemen Serupa: Array adalah kumpulan elemen yang serupa jenisnya. Ini berarti setiap elemen dalam array memiliki tipe data yang sama, seperti integer, float, karakter, atau bahkan tipe data yang lebih kompleks seperti string atau struktur.

Dapat dideklarasikan array bilangan bulat sebagai berikut:

```
int angka[5] = {1, 2, 3, 4, 5};
```

Dalam contoh ini, `angka` adalah array yang terdiri dari lima elemen bertipe int.

Array dapat dideklarasikan dengan menentukan tipe elemennya dan jumlah elemen yang akan disimpan. Proses akses elemen menggunakan indeks, yang dimulai dari 0. Contoh:

```
int nilai = angka[2];  // Mengakses elemen ketiga array, nilai sekarang 3
```

B.Link List

Linked list adalah struktur data linier yang terdiri dari serangkaian simpul (node) yang terhubung satu sama lain dengan menggunakan pointer. Setiap simpul dalam linked list memiliki dua bagian utama: data dan pointer yang menunjukkan ke simpul berikutnya dalam linked list.

Contoh struktur untuk singly linked list adalah sebagai berikut:

```
struct Node {
    int data;
    struct Node* next;
};
```

Dalam penggunaannya, linked list dapat dimanipulasi dengan menambah, menghapus, atau mencari simpul sesuai kebutuhan aplikasi.

1. Node (simpul): Merupakan unit dasar dari linked list. Setiap simpul memiliki dua komponen: data (informasi yang disimpan dalam simpul) dan pointer (alamat memori dari simpul berikutnya dalam linked list).

2. Pointer: Pointer digunakan untuk menghubungkan satu simpul dengan simpul berikutnya dalam linked list. Biasanya, linked list memiliki pointer "next" yang menunjukkan ke simpul berikutnya.
3. Tipe-tipe Linked List:
    a. Singly Linked List: Setiap simpul memiliki satu pointer yang menunjuk ke simpul berikutnya dalam linked list.
    b. Doubly Linked List: Setiap simpul memiliki dua pointer, yaitu pointer "next" yang menunjuk ke simpul berikutnya, dan pointer "prev" yang menunjuk ke simpul sebelumnya.
    c. Circular Linked List: Linked list di mana simpul terakhir menunjuk kembali ke simpul pertama, membentuk lingkaran.

4. Operasi-operasi pada Linked List:
    a. Insertion: Menambahkan simpul baru ke dalam linked list.
    b. Deletion: Menghapus simpul tertentu dari linked list.
    c. Traversal: Mengakses dan memanipulasi setiap elemen dalam linked list.
    d. Searching: Mencari simpul dengan nilai tertentu dalam linked list.
    e. Concatenation: Menggabungkan dua linked list menjadi satu.
    f. Sorting: Mengurutkan linked list berdasarkan nilai data.

C.Double Link List

Double linked list adalah struktur data linier yang terdiri dari serangkaian simpul (node) yang terhubung satu sama lain dengan menggunakan dua pointer, yaitu pointer "next" yang menunjuk ke simpul berikutnya dan pointer "prev" yang menunjuk ke simpul sebelumnya. Dalam double linked list, setiap simpul memiliki dua bagian utama: data (informasi yang disimpan dalam simpul) dan dua pointer (alamat memori dari simpul berikutnya dan sebelumnya dalam linked list).

Contoh sederhana penggunaan double linked list adalah sebagai berikut:

```
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};

// Membuat simpul baru
struct Node* newNode = (struct Node*)malloc(sizeof(struct
Node));
newNode->data = 42;
newNode->next = NULL;
newNode->prev = NULL;
```

1. Node (simpul): Merupakan unit dasar dari double linked list. Setiap simpul memiliki dua komponen: data (informasi yang disimpan dalam simpul) dan dua pointer, yaitu pointer "next" yang menunjuk ke simpul berikutnya, dan pointer "prev" yang menunjuk ke simpul sebelumnya.
2. Pointer "next" dan "prev": Pointer "next" digunakan untuk menghubungkan simpul saat ini dengan simpul berikutnya, sedangkan pointer "prev" digunakan untuk menghubungkan simpul saat ini dengan simpul sebelumnya.
3. Keuntungan Double Linked List:
    a. Dapat bergerak maju dan mundur melalui linked list, sehingga operasi seperti pencarian, penambahan, dan penghapusan bisa lebih efisien.
    b. Memungkinkan akses langsung ke simpul sebelumnya dari simpul tertentu, yang berguna untuk beberapa operasi dan algoritma.
4. Operasi-operasi pada Double Linked List:
    a. Insertion: Menambahkan simpul baru ke dalam double linked list.
    b. Deletion: Menghapus simpul tertentu dari double linked list.
    c. Traversal: Mengakses dan memanipulasi setiap elemen dalam double linked list dengan bergerak maju atau mundur.
    d. Searching: Mencari simpul dengan nilai tertentu dalam double linked list.
    e. Concatenation: Menggabungkan dua double linked list menjadi satu.
    f. Sorting: Mengurutkan double linked list berdasarkan nilai data.

D.Circular Link List

Circular linked list adalah struktur data linier yang mirip dengan linked list biasa, namun dengan perbedaan bahwa simpul terakhir dalam circular linked list menunjuk kembali ke simpul pertama, membentuk lingkaran tertutup. Dengan kata lain, dalam circular linked list, tidak ada simpul yang menunjuk ke nilai NULL, karena simpul terakhir mengarah kembali ke simpul pertama.

Contoh sederhana penggunaan circular linked list adalah sebagai berikut:

```
struct Node {
    int data;
    struct Node* next;
};

// Membuat simpul baru
struct Node* newNode = (struct Node*)malloc(sizeof(struct
Node));
newNode->data = 42;
newNode->next = NULL;  // Perlu diatur saat menghubungkan
dengan simpul lainnya
```

1. Node (simpul): Merupakan unit dasar dari circular linked list. Setiap simpul memiliki dua komponen: data (informasi yang disimpan dalam simpul) dan pointer (alamat memori dari simpul berikutnya dalam linked list).
2. Pointer "next": Pointer "next" digunakan untuk menghubungkan satu simpul dengan simpul berikutnya dalam circular linked list.
3. Simpul terakhir yang menunjuk kembali ke simpul pertama: Ini adalah fitur khas dari circular linked list. Simpul terakhir dalam circular linked list memiliki pointer "next" yang menunjuk kembali ke simpul pertama, membentuk lingkaran.
4. Operasi-operasi pada Circular Linked List:
    a. Insertion: Menambahkan simpul baru ke dalam circular linked list.
    b. Deletion: Menghapus simpul tertentu dari circular linked list.
    c. Traversal: Mengakses dan memanipulasi setiap elemen dalam circular linked list dengan melakukan traversing dari simpul awal ke simpul akhir atau sebaliknya.

DAFTAR PUSAKA

Sjukani, Moh,2012,*Struktur Data (Algoritma dan Struktur Data dengan C,C++*,Jakarta:Mitra Wacana Media

Reek, Kenneth D. "*Pointers on C*." Addison-Wesley Professional, 1997. - Buku inimemberikan pandangan mendalam tentang penggunaan pointer dalam bahasa C dengan contoh kode yang kaya dan penjelasan yang jelas.

Langsam, Yedidyah, Moshe J. Augenstein, and Aaron M. Tenenbaum. "*Data Structures Using C and C++*." Pearson, 2009. - Buku ini memberikan pemahaman yang mendalam tentang struktur data menggunakan bahasa C dan C++, termasuk Linked List, Double Linked List, dan Circular Linked List.