LAPORAN PRAKTIKUM STRUKTUR DATA
MODUL IX

Laporan ini disusun untuk memenuhi Tugas Mata Kuliah

Praktikum struktur data

Disusun Oleh :

M HAMKA ZAINUL ARDHI

NIM: 2311103156

PROGRAM STUDI S1 SISTEM INFORMASI

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

DASAR TEORI

1. Tree

Struktur data tree atau pohon adalah struktur data khusus untuk menyimpan data secara hierarki yang artinya dari satu akar menjadi banyak cabang dari atas ke bawah . ini digunakan untuk mengatur dan menyimpan data di komputer agar dapat digunakan lebih efektif. Ini terdiri dari

1. node pusat, node.

   Node pusat atau "akar" adalah node paling atas dalam pohon. Node ini adalah titik awal atau asal dari pohon dan tidak memiliki parent node (induk).
2. node struktural

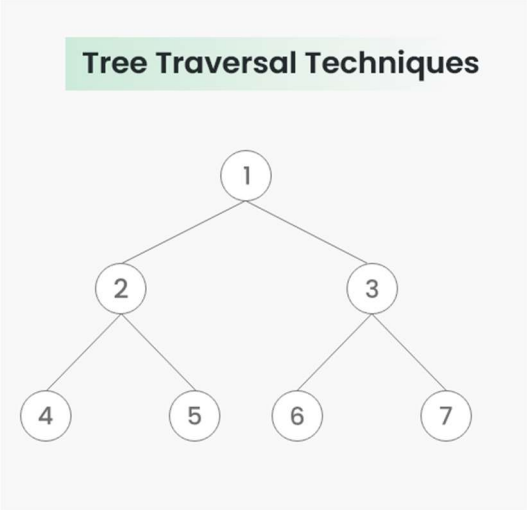
   Node struktural adalah node yang berada di antara akar dan daun. Ini bisa memiliki child nodes (anak) dan juga parent nodes. Node struktural membantu membentuk cabang dan menentukan struktur keseluruhan pohon.
3. sub-node

   Sub-node atau anak adalah node yang memiliki parent node di atasnya. Setiap node bisa memiliki nol atau lebih sub-node.

Hal hal ini membuktikan bahwa struktur data pohon memiliki akar, cabang, dan daun yang terhubung satu sama lain untuk menciptakan struktur data yang efisen. Setiap pohon memiliki satu node pusat yang disebut akar, yang menjadi titik awal dari mana semua node lainnya bercabang. Node dalam pohon bisa memiliki beberapa anak (sub-node) dan tepat satu induk (kecuali untuk akar, yang tidak memiliki induk). Node yang tidak memiliki anak disebut daun. Struktur ini tidak memiliki siklus, artinya tidak ada jalur yang menghubungkan kembali ke node yang sama melalui tepi yang berbeda

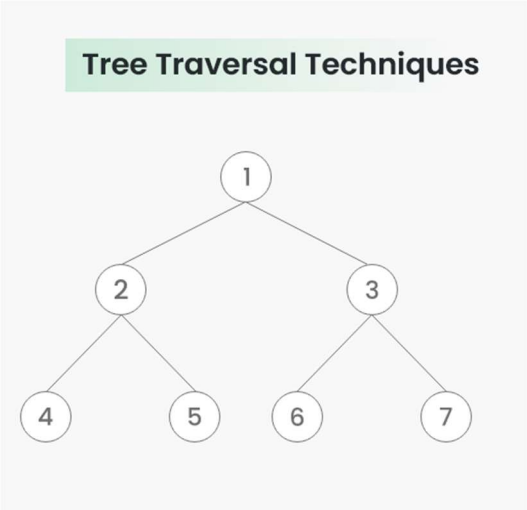
2. Logika dasar struktur Pohon
   dalam struktur data pohon adalah proses mengunjungi setiap node dalam pohon secara sistematis. Terdapat tiga metode utama penelusuran pohon biner: in-order, pre-order, dan post-order, yang masing-masing memiliki aturan dan kegunaan khusus dalam memahami dan mengelola data dalam pohon.

   a) In order
      Rumus = root – left (print) – right



      Hasil = 4 2 5 1 6 3 7

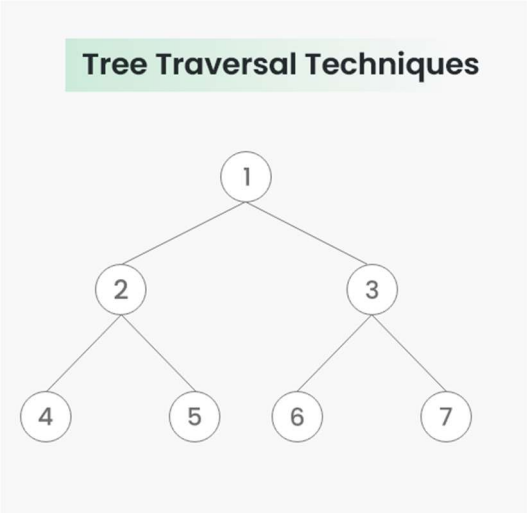
   b) Pre order
      Rumus = root (print) – left – right



      Hasil = 1 2 4 5 3 6 7

c) Post order

Rumus = root – left – right (print)



Hasil = 4 5 3 6 7 3 1

3. Implementasi tree dalam pemograman c++

Membuat dasar dari code struktur data tree dalam suatu program C++ disini menggunakan struct yg memiliki 2 buah pointer, seperti halnya double linked list. Sehingga akan di awali dengan pendeklarasian pointer terlebih dahulu

Source code

```
struct pohon{
    pohon* kanan;
    char data;
    pohon* kiri;
};

pohon* simpul;
```

GUIDED

1. Membuat program untuk membuat struktur dasar data dalam binary tree. Menjalankan program nya apakah berfunsi dengan semestinya lalu akan di analisis

```
#include <iostream>
using namespace std;

struct pohon
{
    pohon* kanan;
    pohon* kiri;
    char data;
};

pohon* simpul;
pohon* root;
pohon* saatini;
pohon* alamat[256];

// inisialisai root
void inisialisasi(){
    root = NULL;
}

//manambahakan data//
 void simpulbaru(char datamasukan) {
    simpul = new pohon;
    simpul->data = datamasukan;
    simpul->kanan = NULL;
    simpul->kiri = NULL;
 }
//manambahkan akar
void simpulakar () {
    if (root == NULL){
        char dataanda;
        cout << "silahkan masukan data :";
        cin >> dataanda;
        simpulbaru(dataanda);
        root = simpul;
        cout << "root terbentuk" <<endl;
    } else {
        cout << "root sudah ada" <<endl;
    }
}

void tambahsimpul () {
    if (root=NULL){
    int i = 1;
```

```
    int j = 1;
    int penanda =0;
    char datauser;
    alamat[i] = root;
    while (penanda == 0 && j < 256) {
        cout <<"masukan data kiri" << endl;
        cin >> datauser;
        if (datauser !='0') {
            simpulbaru(datauser);
            saatini = alamat[i];
            saatini->kiri=simpul;
            j++;
            alamat[i] = simpul;
        } else {
            penanda = 1;
            j++;
            alamat[j] = NULL;
        }
        if (penanda == 0){
            cout<< "masukan data kanan : " ;
            cin >> datauser;
            if(datauser !='0'){
                simpulbaru(datauser);
                saatini = alamat[i];
                saatini->kanan = simpul;
                j++;
                alamat[j] = simpul;
            } else {
                penanda = 1;
                j++;
                alamat[j] = NULL;
            }
          }
          i++;
       }
    }
}
void bacaPohon() {
    if (root != NULL) {
        int i = 1, n = 1, pencacah = 0;
        cout << endl;

        while (alamat[i] != NULL) {
            saatini = alamat[i];
            cout << saatini->data << " ";
            pencacah++;
            if (pencacah == n) {
                cout << endl;
```

```
                pencacah = 0;
                n = n * 2;
            }
            i++;
        }
    }
}

int main() {
    simpulakar();
    tambahsimpul();
    bacaPohon();
    return 0;
}
```

Disini langkah pertama ialah mendefinisikan simpul/node dari binary tree dengan tiga anggota terdapat kanan, kiri dan data, data untuk menyimpan karakter, lalu mendefinisian pointer pointer guna untuk menyimpan pinter ke simpul simpul yang ada pada binary tree dan sete;lah nya dideklarasikanlah fungsi bertipe data void ada beberapa fungsi di program ini antara lain

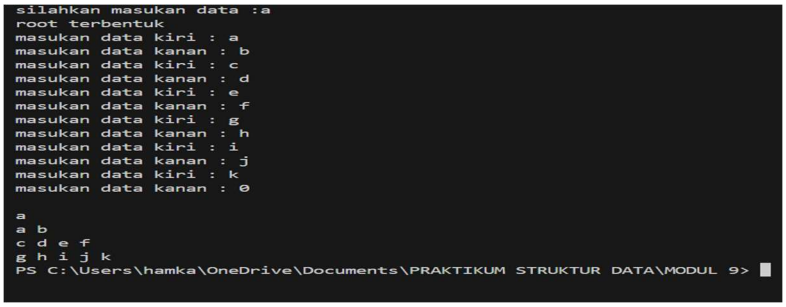
‘fungsi menambahkan simpul akar’ yang berfungsi memeriksa pohon apakah pohon tersebut sudah memiliki akar jika belum user akan di minta untuk memasukan data atau memnuat simpul baru untuk menetapkannya sebagai akar pohon

‘fungsi menambahkan simpul’ disini pengguna di minta untuk memasukan data untuk simpul bagian kiri dan bagian kanan dari setiap simpul yang ada

‘fungsi baca pohonn’ atau menampilkan data dari data setiap simpul dicetak kelayar dan ini adalah fungsi yang menjadi output dari program ini

Dan di fungsi main adalah fungsi utama disini setiap fungi fungsi lain nya di dekralasikan kembali untuk di jadikan output program Program ini membantu dalam memahami bagaimana sebuah binery tree dapat dibangun, diubah, dan dibaca menggunakan fungsi-fungsi dasar di atas.

Hasil Program



UNGUIDED

1. Modifikasi guided di atas agar dapat melakukan penyisipan simpul pada lokasi dimanapun, sehingga bisa menysesuaikan dengan keinginan pengguna.

Source code

```
#include <iostream>
using namespace std;

struct pohon
{
    pohon* kanan;
    pohon* kiri;
    char data;
};

pohon* simpul;
pohon* root;
pohon* saatini;
pohon* alamat[256];

// inisialisai root
void inisialisasi(){
    root = NULL;
}

//manambahakan data//
 void simpulbaru(char datamasukan) {
    simpul = new pohon;
    simpul->data = datamasukan;
    simpul->kanan = NULL;
    simpul->kiri = NULL;
 }
//manambahkan akar
void simpulakar () {
    if (root == NULL){
        char dataanda;
        cout << "silahkan masukan data :";
        cin >> dataanda;
        simpulbaru(dataanda);
        root = simpul;
        cout << "root terbentuk" <<endl;
    } else {
        cout << "root sudah ada" <<endl;
    }
}
```

```
void tambahsimpulkiri () {
    if (root !=NULL){
    int i = 1;
    int j = 1;
    int penanda =0;
    char datauser;
    alamat[i] = root;
    while (penanda == 0 && j < 256) {
        cout <<"masukan data kiri : " ;
        cin >> datauser;
        if (datauser !='0') {
            simpulbaru(datauser);
            saatini = alamat[i];
            saatini->kiri=simpul;
            j++;
            alamat[j] = simpul;
        } else {
            penanda = 1;
            j++;
            alamat[j] = NULL;
        }
      }
    }
}
            void tambahsimpulkanan () {
            int i = 1;
            int j = 1;
            int penanda =0;
            char datauser;
            if (penanda == 0){
            cout<< "masukan data kanan : " ;
            cin >> datauser;
            if(datauser !='0'){
                simpulbaru(datauser);
                saatini = alamat[i];
                saatini->kanan = simpul;
                j++;
                alamat[j] = simpul;
            } else {
                penanda = 1;
                j++;
                alamat[j] = NULL;
            }
          }
          i++;
        }
```

```
void bacaPohon() {
    if (root != NULL) {
        int i, n, pencacah;
        i = 1;
         n = 1;
         pencacah = 0;
        cout << endl;

        while (alamat[i] != NULL) {
            saatini = alamat[i];
            cout << saatini->data << " ";
            pencacah++;
            if (pencacah == n) {
                cout << endl;
                pencacah = 0;
                n = n * 2;
            }
            i++;
        }
    }
}

int main() {
inisialisasi();
    int pilih;
    do {
        cout << endl;
        cout << "MENU" <<endl;
        cout << "1. membuat simpul akar" <<endl;
        cout << "2. menambahkan simpul kiri" <<endl;
        cout << "3. menambahkan simpul kanan" <<endl;
        cout << "4. tampilkan pohon" <<endl;
        cout << "5. keluar" <<endl;
        cout << "pilih : ";
        cin >> pilih ;
        cout <<endl;

        switch (pilih) {
            case 1:
            simpulakar ();
            break;

            case 2 :
            tambahsimpulkiri ();
            break;

            case 3 :
            tambahsimpulkanan ();
```

```
            break;

            case 4 :
            bacaPohon();
            break;

            case 5 :
                cout << "Keluar dari program" << endl;
                return 0;
            default:
                cout << "Pilihan tidak valid" << endl;
                break;

        }
    } while (true);

}
```

Disini hanya penambahan menu pada program guided sehingga user bisa memilih melakukan penyisipan simpul pada lokasi dimanapun,

Pertama tama yang paling utama adalah pendeklarasian varibel pilih dengan tipe data ‘int’ ini memungkin kan user memilih menu berdasar kan nomor nya saja, selanjut nya adalah pendeklarasian fungsi inisialisasi (); berguna untuk menyimpan data yang dimasukkan oleh pengguna yang kemudian akan ditambahkan ke fungsi.

Struktur ‘swicth case’ berguna untuk melakukan pengambilan keputusan berdasarkan nilai dari yang di inputkan pengguna. switch-case memungkinkan user untuk memeriksa nilai dari suatu ekspresi dan menjalankan blok kode operasi yang sesuai dengan nilai tersebut. Setiap ‘case’ mengikuti ekspresi dan diikuti oleh nilai yang akan dibandingkan dengan ekspresi tersebut. Jika nilai ekspresi cocok dengan nilai yang tercantum dalam suatu case, blok kode yang terkait dengan case tersebut akan dieksekusi.

Kemudian, terdapat pernyataan ‘break’ yang memiliki fungsi untuk menghentikan eksekusi lebih lanjut dari switch-case dan keluar dari blok switch. Pernyataan break memastikan bahwa setelah satu case dieksekusi, eksekusi akan keluar dari blok switch dan tidak melanjutkan ke case berikutnya.

Opsionalnya, ada blok default yang akan dijalankan jika tidak ada nilai case yang cocok dengan nilai ekspresi yang dievaluasi. default memberikan kemungkinan untuk menangani situasi di mana tidak ada kecocokan dengan nilai-nilai yang telah ditentukan dalam case sebelumnya.

1. Hasil Menu 1 membuat simpul akar

```
MENU
1. membuat simpul akar
2. menambahkan simpul kiri
3. menambahkan simpul kanan
4. tampilkan pohon
5. keluar
pilih : 1

silahkan masukan data :a
root terbentuk
```

2. Hasil Menu 2 dan 3 menambahkan data kiri dan masukan data kanan

```
MENU
1. membuat simpul akar
2. menambahkan simpul kiri
3. menambahkan simpul kanan
4. tampilkan pohon
5. keluar
pilih : 2

masukan data kiri : a
masukan data kiri : b
masukan data kiri : 0
```

```
MENU
1. membuat simpul akar
2. menambahkan simpul kiri
3. menambahkan simpul kanan
4. tampilkan pohon
5. keluar
pilih : 3

masukan data kanan : v
```

3. Hasil menu 4 Tampilkan Pohon

```
MENU
1. membuat simpul akar
2. menambahkan simpul kiri
3. menambahkan simpul kanan
4. tampilkan pohon
5. keluar
pilih : 4


a
v b
```

4. Hasil Menu 5 Keluar

```
MENU
1. membuat simpul akar
2. menambahkan simpul kiri
3. menambahkan simpul kanan
4. tampilkan pohon
5. keluar
pilih :
5

Keluar dari program
```

KESIMPULAN

Struktur data pohon adalah model hierarkis yang terdiri dari node-node yang terhubung oleh tepi, dengan satu node sebagai akar dan node lainnya sebagai sub-node yang bercabang dari akar. Pohon tidak memiliki siklus dan memungkinkan penelusuran melalui tiga metode utama: in-order, pre-order, dan post-order. merepresentasikan hubungan hierarkis secara efisien dan terstruktur.

REFRENSI

1. MODUL 9 STRUKTUR DATA
2. https://www.simplilearn.com/tutorials/data-structure-tutorial/trees-in-data-structure
3. https://www.geeksforgeeks.org/tree-traversal-techniques-in-python/
4. https://www.programiz.com/dsa/trees