LAPORAN PRAKTIKUM

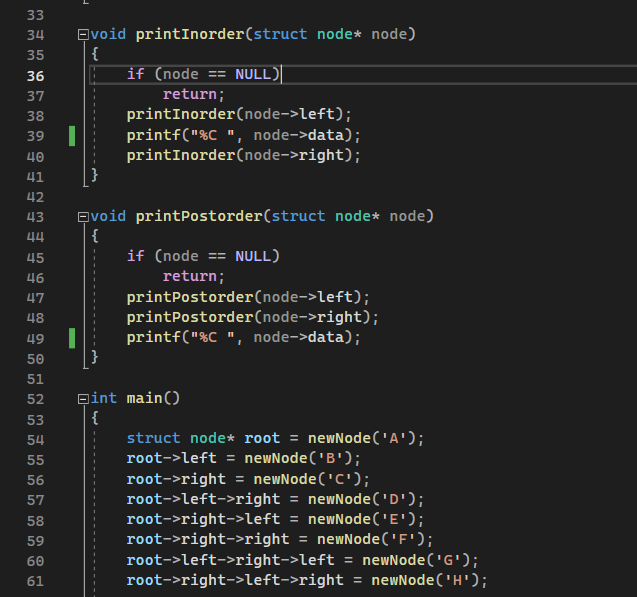
|  |  |
| --- | --- |
| Matakuliah | Struktur Data |
| Pertemuan ke | 10 |
| Nama Praktikan | Wijayanto Agung Wibowo |
| NIM | 22.11.4552 |
| NILAI (diisi oleh dosen / asisten praktikum) |  |

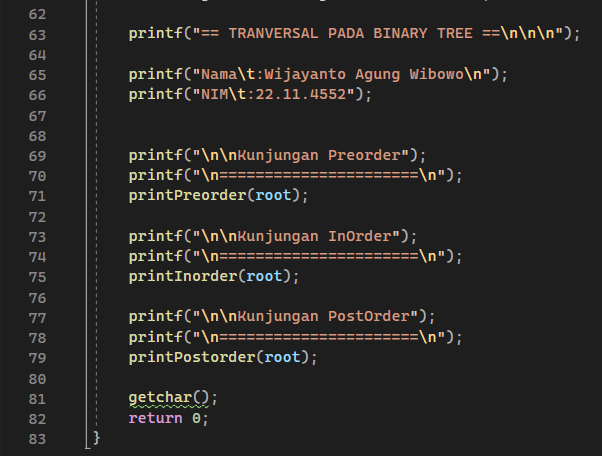
1. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

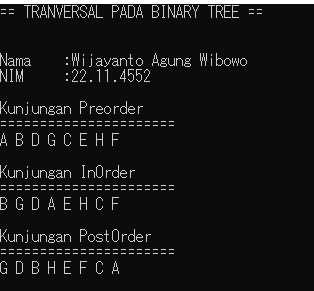
1. Memahami tipe data dasar Tree
2. Memahami tipe data bentukan Tree
3. Stuktur Program Tree menggunakan bahasa C++ Praktikum
4. Hasil Percobaan
   1. Praktikum
5. Tampilan Coding







1. Hasil Running



1. Penjelasan

Pada program ini dideklarasikan sebuah struct yang didalamnya terdapat tiga field,

typedef struct Node{

int data;

Node \*left;

Node \*right;

};

- Field pertama: variabel bertipe data integer, digunakan untuk menyimpan data pada node.

- Field kedua: variable pointer yang bertipe data abstrak (NODE), digunakan untuk menunjuk node anak(child) sebelah kiri.

- Field ketiga: variabel pointer yang bertipe data abtrak (NODE), digunakan untuk menunjuk node anak(child) sebelah kanan.

# Pada Tree terdapat operasi Traverse yaitu operasi kunjungan terhadap node-node dalam pohon dimana masing-masing node akan dikunjungi sekali. Dalam program ini digunakan tiga tranverse, yakni :

PreOrder: node yang dikunjungi (induk), kunjungi left, kunjungi right

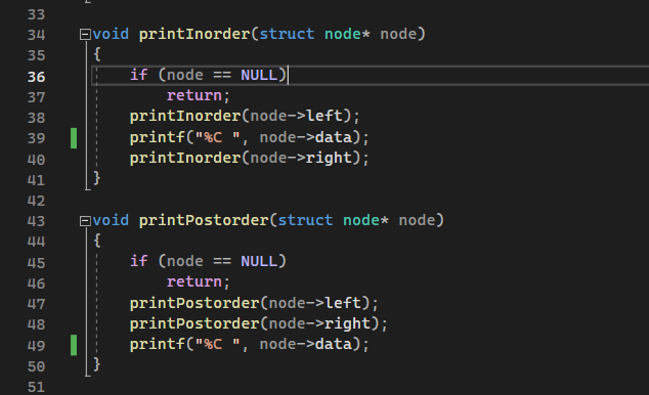
InOrder: kunjungi left, node yang dikunjungi (induk), kunjungi right

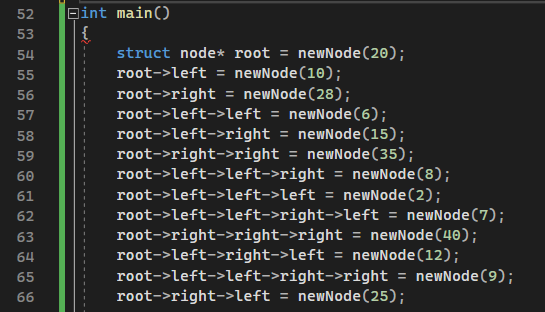
PostOrder: kunjungi left, kunjungi right, node yang dikunjungi (induk)

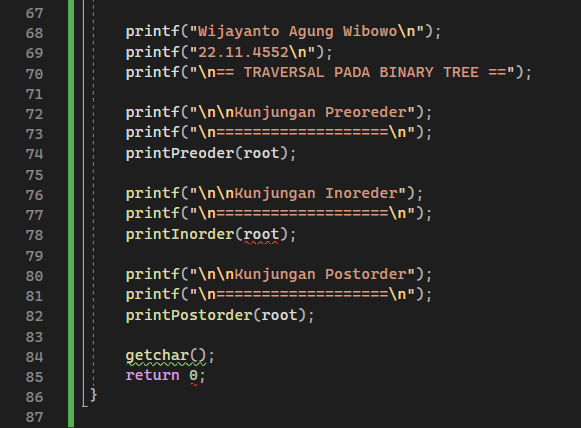
* 1. Latihan 1

1. Tampilan Coding

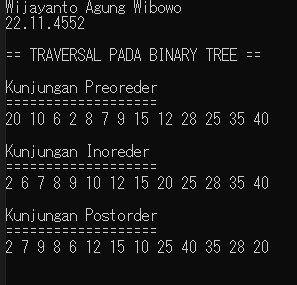








1. Hasil Running



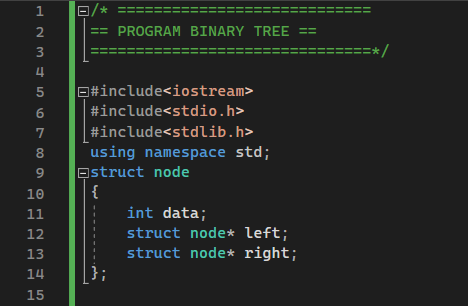
1. Penjelasan

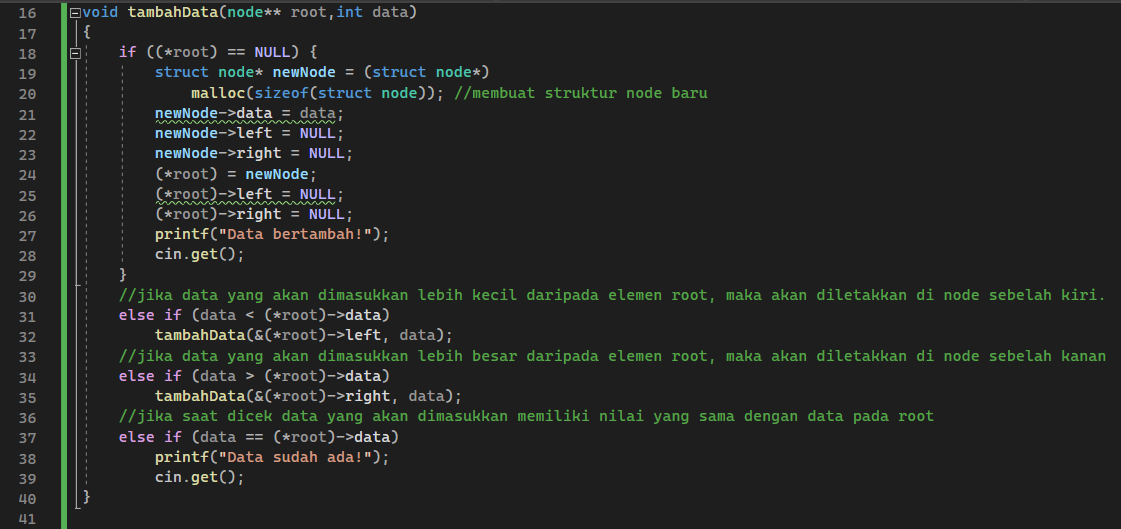
Penginputan data tree bisa dengan manual. Tetapi terdapat kaidah tertentu, yang mengharuskan pengisian data jika data yang dimasukan itu kurang dari root, maka data harus dimasukan ke child sebelah kiri. Namun apabila data lebih besar dari root, maka data harus dimasukan ke dalam child sebelah kanan.

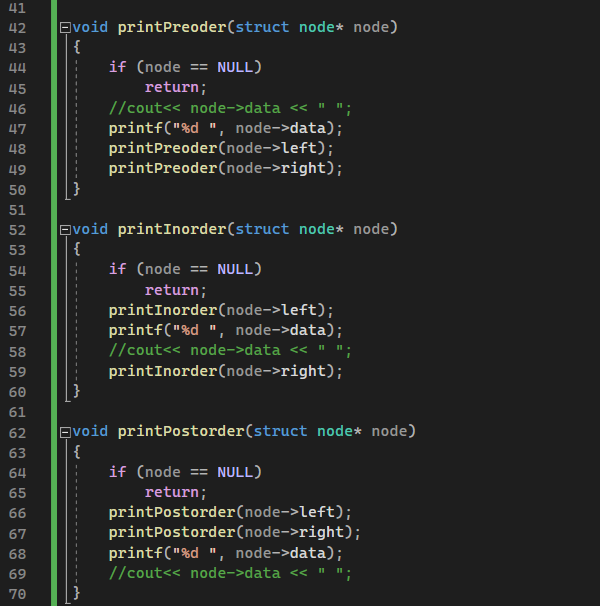
Untuk melihat data yang di sort, bisa menggunakan kunjungan inorder.

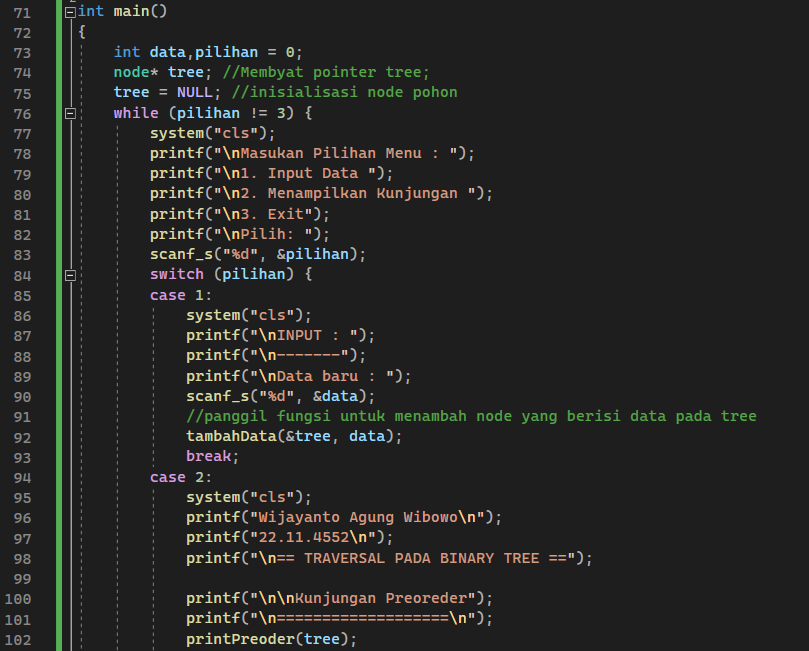
* 1. Challenge

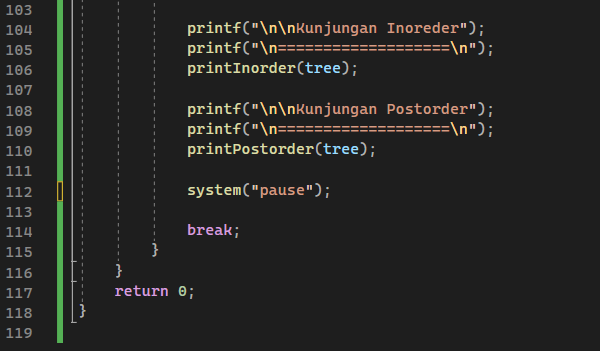
1. Tampilan Coding



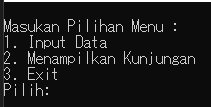




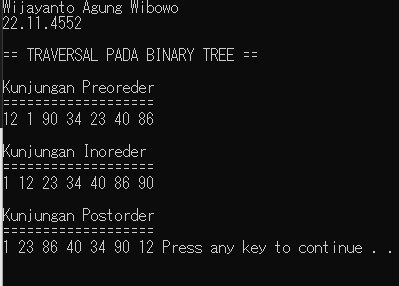




1. Hasil Running



Hasil setelah Menginput data 12,90,86,1,34,23,40 Lalu di tampilkan Kunjungan:



1. Penjelasan

Fungsi tambahData menerima dua parameter: sebuah double pointer ke akar pohon pencarian biner (node\*\* root) dan nilai data yang akan dimasukkan (int data).

Fungsi ini memulai dengan memeriksa apakah pointer ke root ((\*root)) adalah NULL, yang menandakan bahwa pohon masih kosong. Jika pohon kosong, sebuah node baru dibuat menggunakan alokasi memori dinamis (malloc) dan nilai data yang diberikan ditetapkan pada node baru tersebut. Pointer kiri dan kanan dari node baru diatur menjadi NULL, dan pointer root diperbarui agar menunjuk ke node baru tersebut. Pesan "Data bertambah!" dicetak, dan cin.get() digunakan untuk menjeda program hingga pengguna menekan tombol Enter.

- Jika pohon tidak kosong, fungsi membandingkan nilai data dengan nilai data dari node saat ini ((\*root)->data) untuk menentukan posisi yang tepat untuk penyisipan.

-Jika data kurang dari (\*root)->data, maka fungsi tambahData akan dipanggil secara rekursif dengan pointer ke anak kiri sebagai parameter.

- Jika data lebih besar dari (\*root)->data, maka fungsi tambahData akan dipanggil secara rekursif dengan pointer ke anak kanan sebagai parameter.

- Jika data sama dengan (\*root)->data, pesan "Data sudah ada!" akan dicetak, dan cin.get() digunakan untuk menjeda program.

Fungsi ini bertujuan untuk membangun sebuah pohon pencarian biner dengan mengatur data secara terurut. Node-node dengan nilai lebih kecil ditempatkan di sebelah kiri, sedangkan node-node dengan nilai lebih besar ditempatkan di sebelah kanan.

1. Kesimpulan

Setelah melakukan percobaan pada Latihan 1 sampai challenge, saya dapat memahami pengimplementasian binary tree.   
binary tree hanya bisa tercipta oleh node.

1. Referensi (optional)