

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL 9
ADT Binary Tree**

**DISUSUN OLEH :
AJI KARTIKO HARTANTO - 2350081062**



**PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS DAN INFORMATIKA
UNIVERSITAS JENDERAL ACHMAD YANI
TAHUN 2024**

DAFTAR ISI

| | |
|-----------------------------------|----|
| DAFTAR GAMBAR | ii |
| BAB I. HASIL PRAKTIKUM..... | 1 |
| I.1 Latihan..... | 1 |
| I.1.A. Source Code bTree.h..... | 1 |
| I.1.B. Source Code bTree.c | 3 |
| I.1.C. Source Code mBTree.c | 7 |
| I.1.D. Hasil | 9 |
| I.1.E. Analisa | 9 |
| BAB II. KESIMPULAN | 10 |

DAFTAR GAMBAR

| | |
|---|---|
| Gambar I.1 Output Program mBTree.c..... | 9 |
|---|---|

BAB I. HASIL PRAKTIKUM

I.1 Latihan

I.1.A. Source Code bTree.h

```
/*
    Program          : bTree.h
    Author           : 2350081062, Aji Kartiko Hartanto
    Kelas            : C
    Deskripsi         : Header file dari prototype binary tree
    Tanggal          : 12 - 06 - 2024
*/

// preprosesor
#include<conio.h>
#include<stdlib.h>
#include<stdio.h>

/* Deklarasi Btree */
#define nil NULL
#define info(P) (P)->info
#define left(P) (P)->left
#define right(P) (P)->right

//pembentukan tipe BTree
typedef struct tNode *address;
typedef struct tNode{
    int info;
    address left;
    address right;
}Node;
typedef address Tree;

//prototype BinTree
//Kelompok konstruktor
void CreateTree(Tree *T, int x, Tree L, Tree R);
/*
    I.S : T terdefinisi sembarang
    F.S : menghasilkan sebuah pohon T
          menghasilkan sebuah pohon T dari x, L, R jika alokasi berhasil,
```

```

        menghasilkan pohon T = nil jika alokasi gagal
    */

    address Alokasi(int X);
    /* mengirim sebuah address dari x berupa node */

    void DeAlokasi(address P);
    /* merelease memori bernilai P */

    //Kelompok selektor
    int getRoot(Tree T);
    /* mengirim nilai root pada T */

    Tree getLeft(Tree T);
    /* mengirim anak kiri dari T */

    Tree getRight(Tree T);
    /* mengirim anak kanan dari T */

    int TinggiPohon(Tree T);
    /* mengirim tinggi pohon, pohon kosong = 0 */

    int Maksimum(int Kiri, int Kanan);
    /* mengirim nilai terbesar dari Kiri atau Kanan */

    //Operasi I/O
    Tree InsSearch(Tree T, int x);
    /* menghasilkan sebuah pohon dengan nilai X */

    void CetakTree(Tree T);
    /*
        I.S : T terdefinisi sembarang tidak kosong
        F.S : semua simpul dari T tercetak
    */

    void PreOrder(Tree T);
    /*
        I.S : T terdefinisi sembarang
        F.S : semua simpul T diproses secara preorder :
    */

```

```

                                akar - kiri - kanan
*/

void InOrder(Tree T);
/*
    I.S : T terdefinisi sembarang
    F.S : semua simpul T diproses secara inorder :
            kiri - akar - kanan
*/

void PostOrder(Tree T);
/*
    I.S : T terdefinisi sembarang
    F.S : semua simpul T diproses secara postorder 2
            kiri - kanan - akar
*/

```

I.1.B. Source Code bTree.c

```

/*
    Program      : bTree.c
    Author       : 2350081062, Aji Kartiko Hartanto
    Kelas        : C
    Deskripsi    : Body file dari prototype binary tree
    Tanggal      : 12 - 06 - 2024
*/

#include "bTree.h"
#include <stdio.h>
#include <conio.h>

/* Realisasi dari Prototype binary tree */
//Kelompok konstruktor
void CreateTree(Tree *T, int x, Tree L, Tree R){
/* I.S : T terdefinisi sembarang
    F.S : menghasilkan sebuah pohon T
    menghasilkan sebuah pohon T dari x, L, R jika alokasi berhasil,
    menghasilkan pohon T = nil jika alokasi gagal

```

```

*/

    // kamus

    // algoritma
    *T = Alokasi(x);
    if(*T != nil){
        left(*T) = L;
        right(*T) = R;
    }
}

address Alokasi(int X){
/* mengirim sebuah address dari x berupa node */
    // kamus
    address P;

    // algoritma
    P = (address) malloc(sizeof(Node));
    if(P!=nil){
        info(P) = X;
        left(P) = nil;
        right(P) = nil;
    }
    return(P);
}

void DeAlokasi(address P){
/* merelease memori bernilai P */
    free(P);
}

//Kelompok selektor
int getRoot(Tree T);
/* mengirim nilai root pada T */
Tree getLeft(Tree T);
/* mengirim anak kiri dari T */
Tree getRight(Tree T);
/* mengirim anak kanan dari T */
int TinggiPohon(Tree T){
/* mengirim tinggi pohon, pohon kosong = 0 */

```



```

//kamus
int tinggi;

//algoritma
tinggi = 0;
if(T!=nil){
    tinggi=1+Maksimum(TinggiPohon(left(T)),TinggiPohon(right(T));
}
return(tinggi);
}

int Maksimum(int Kiri, int Kanan){
/* mengirim nilai terbesar dari Kiri atau Kanan */
    //kamus
    //algoritma
    if(Kiri > Kanan)
        return(Kiri);
    else
        return(Kanan);
}

//Operasi I/O
Tree InsSearch(Tree T, int x){
/* menghasilkan sebuah pohon dengan nilai x */
    //kamus
    //algoritma
    if(T == nil)
        CreateTree(&T,x,nil,nil);
    else if(x < info(T))
        left(T) = InsSearch(left(T), x);
    else
        right(T) = InsSearch(right(T), x);
    return(T);
}

void CetakTree(Tree T){
/* I.S : T terdefinisi sembarang tidak kosong
   F.S : semua simpul dari T tercetak
*/
    //kamus

```

```

        //algoritma
        if(T != nil){
            printf("%d", info(T));
        }
    }
}

void PreOrder(Tree T){
/* I.S : T terdefinisi sembarang
F.S : semua simpul T diproses secara preorder :
akar - kiri - kanan
*/

    //kamus
    //algoritma
    if(T!=nil){
        CetakTree(T);
        PreOrder(left(T));
        PreOrder(right(T));
    }
}

void InOrder(Tree T){
/* I.S : T terdefinisi sembarang
F.S : semua simpul T diproses secara inorder :
kiri - akar - kanan
*/

    //kamus
    //algoritma
    if(T!=nil){
        PreOrder(left(T));
        CetakTree(T);
        PreOrder(right(T));
    }
}

void PostOrder(Tree T){
/* I.S : T terdefinisi sembarang
F.S : semua simpul T diproses secara postorder :
kiri - kanan - akar
*/

    //kamus

```

```

//algoritma
if (T!=nil) {
    PreOrder(left(T));
    PreOrder(right(T));
    CetakTree(T);
}
}

```

I.1.C. Source Code mBTree.c

```

/*
    Program      : mBTree.c
    Author       : 2350081062, Aji Kartiko Hartanto
    Kelas        : C
    Deskripsi    : Main Driver binary tree
    Tanggal     : 12 - 06 - 2024
*/

#include "bTree.h"
#include <stdio.h>
#include <stdlib.h>

int main() {
    // Kamus global
    Tree MyPohon = nil;
    int N, akar;

    // Algoritma
    // Hapus layar tergantung pada sistem operasi
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif

    printf("Masukkan akar pohon: ");
    scanf("%d", &akar);
    MyPohon = Alokasi(akar);

    printf("Masukan Bilangan: ");

```

```

scanf("%d", &N);
while (N != 9999) {
    InsSearch(MyPohon, N);
    printf("Masukan Bilangan: ");
    scanf("%d", &N);
}

printf("PreOrder: ");
PreOrder(MyPohon);
printf("\n");

printf("InOrder: ");
InOrder(MyPohon);
printf("\n");

printf("PostOrder: ");
PostOrder(MyPohon);
printf("\n");

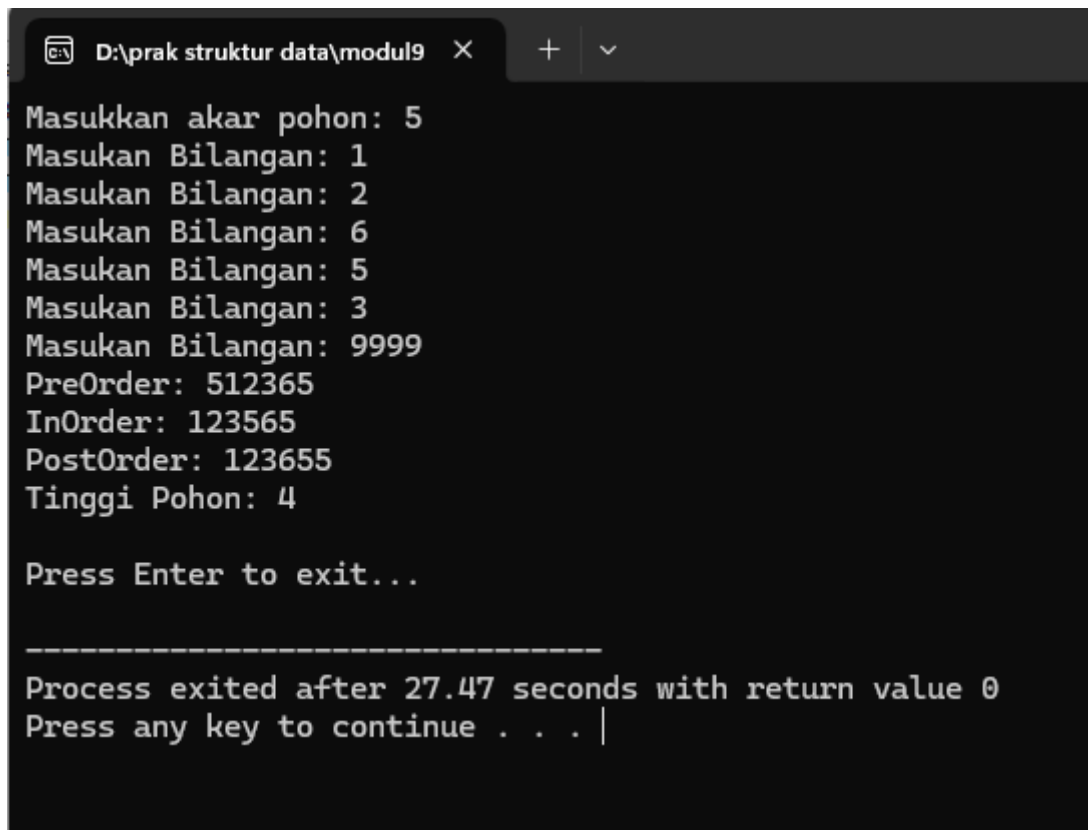
printf("Tinggi Pohon: %d\n", TinggiPohon(MyPohon));

// Menunggu input sebelum keluar, agar konsol tidak langsung tertutup
printf("\nPress Enter to exit...");
getchar(); // Mengambil input sisa dari scanf sebelumnya
getchar(); // Menunggu input sebenarnya

return 0;
}

```

I.1.D. Hasil



```
D:\prak struktur data\modul9 X + v
Masukkan akar pohon: 5
Masukan Bilangan: 1
Masukan Bilangan: 2
Masukan Bilangan: 6
Masukan Bilangan: 5
Masukan Bilangan: 3
Masukan Bilangan: 9999
PreOrder: 512365
InOrder: 123565
PostOrder: 123655
Tinggi Pohon: 4

Press Enter to exit...

-----
Process exited after 27.47 seconds with return value 0
Press any key to continue . . . |
```

Gambar I.1 Output Program mBTree.c

I.1.E. Analisa

Tiga file yang termasuk dalam program mengimplementasikan dan menunjukkan penggunaan Jenis Data Abstract (ADT) Pohon Biner secara bersamaan. File header `bTree.h` mendefinisikan struktur data dan fungsi prototipe yang digunakan untuk mengelola pohon biner. Dalam file implementasi, fungsi-fungsi ini melakukan hal-hal seperti alokasi dan dealokasi memori, penambahan node, dan berbagai metode traversal (PreOrder, InOrder, dan PostOrder). Fungsi-fungsi ini menggunakan definisi pohon biner untuk membuat dan mengubah struktur data pohon. Kemudian, program utama menggabungkan semuanya dengan meminta pengguna untuk memasukkan nilai yang akan dimasukkan ke dalam pohon, mencetak hasil traversal pohon, dan menghitung tinggi pohon. Program ini menunjukkan bagaimana operasi dasar dilakukan pada pohon biner, serta bagaimana data dalam pohon dapat diakses dan ditampilkan dalam berbagai urutan.

BAB II. KESIMPULAN

Pada praktikum modul 9 ini, kita telah mempelajari adt binary tree. Program ini menunjukkan pentingnya ADT Binary Tree dalam pengelolaan data yang terstruktur dan bagaimana operasi dasar pada pohon biner dilakukan, memberikan dasar yang kuat untuk pengembangan dan pemahaman lebih lanjut tentang struktur data yang lebih kompleks. Dengan memahami pohon biner, kita dapat mengimplementasikan berbagai algoritma pencarian dan pengurutan secara efisien, serta menyelesaikan banyak masalah yang memerlukan pengelolaan data hierarkis.