16TIN2074 – Struktur Dasar Algoritma
Studi Kasus Non Binary Tree

Disusun Oleh :

Kelompok 6 - 1B D4 Teknik Informatika

Arsal Fadilah	201524036
Faizal Abdul Hakim	201524043
Fiora Berliana Putri	201524045

Program Studi D4 Teknik Informatika

Jurusan Teknik Komputer dan Informatika

Politeknik Negeri Bandung

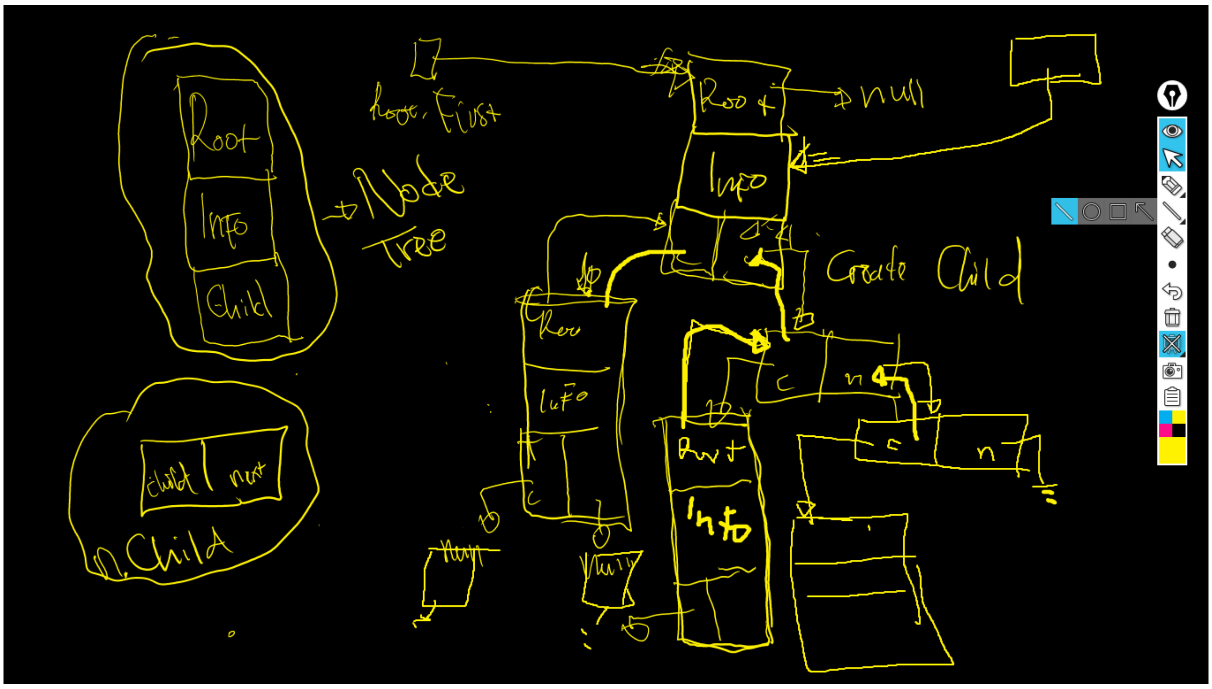
2020/2021

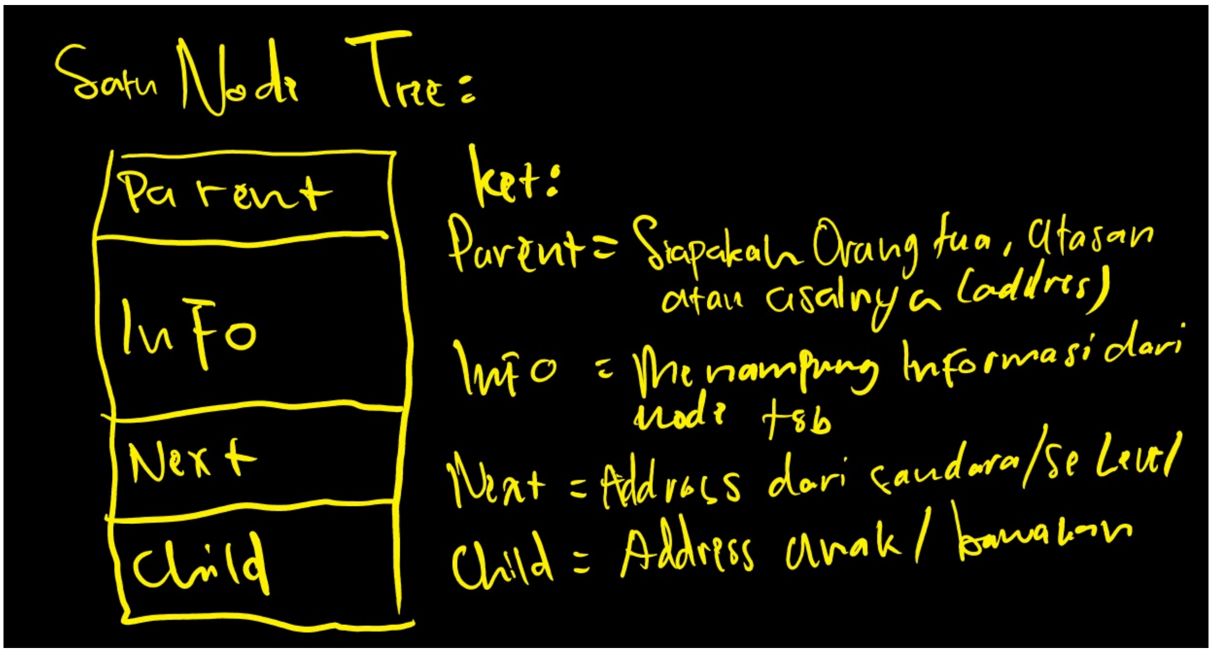
Deskripsi Tugas

Kami diminta untuk membuat sebuah rancangan aplikasi penelusuran jabatan. Fitur umum yang harus ada dalam aplikasi ini adalah penambahan jabatan beserta pejabatnya, dan fitur utama yang harus ada adalah penelusuran jabatan, yaitu menelusuri siapa atasan atau bawahan dari yang dicari.

Tugas

Visualisasi :





Satu Node Tree:

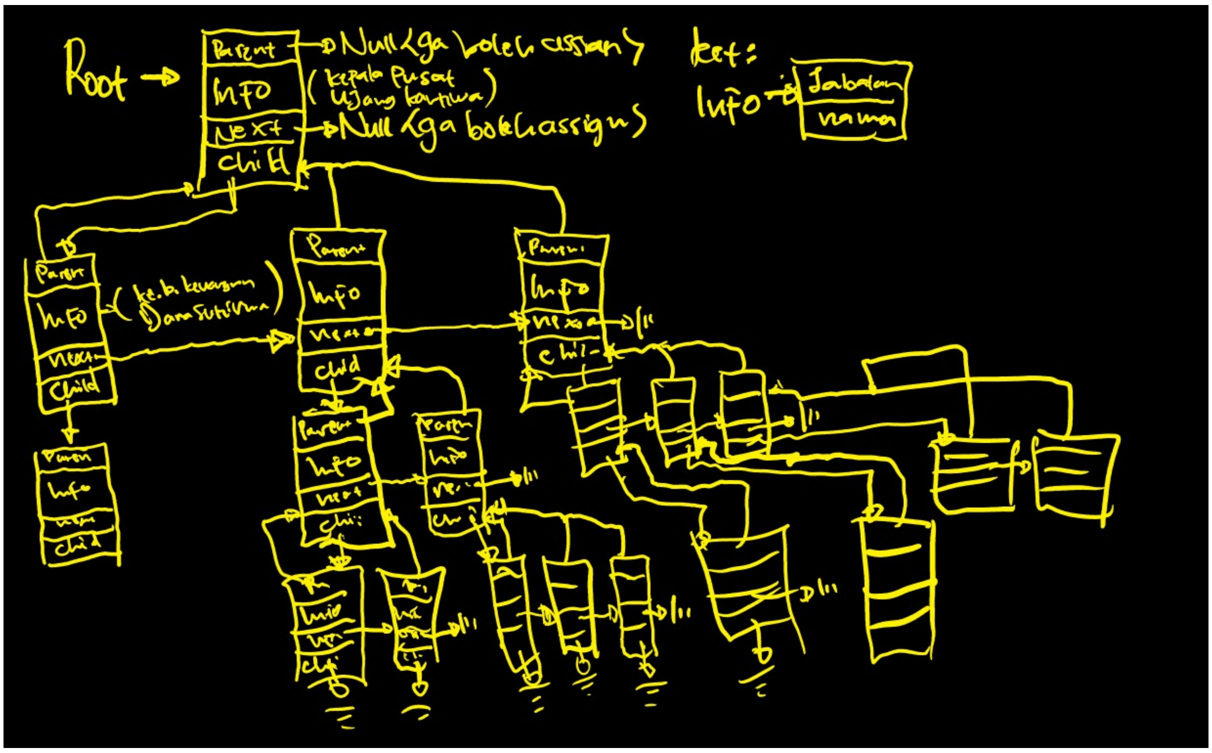
ket:

Parent = Siapakah Orang tua, atasan atau asalnya (address)

Info = Menampung Informasi dari node tsb

Next = Address dari saudara/se Level

Child = Address anak / bawahan



Header :

```
/*** Data Structure Non Binary Tree ***/
/**************************************/

#ifndef Nonbinarytree_H
#define Nonbinarytree_H

#include <stdbool.h>
/*** User Defined Data Type ***/
/** Infotype is info of each node tree **/
/**| If u want edit info only in here |**/
typedef struct
{
    /* data */
    char Nama[100];
    char Jabatan[100];
} infotype;
typedef struct Node *addrNodeTree;

/** Address of node tree **/
typedef addrNodeTree addrParent;
/**
 * Node Tree is a
 * element will be created for
 * ADT non binary tree
 * **/
typedef struct Node
{
    /* data */
    addrParent parents;
    infotype info;
    addrNodeTree child;
    addrNodeTree next;
} NodeTree;

/** nbTree is Non Binary Tree data type. this is will be acces root
(first) for a moment **/
typedef struct
{
    /* data */
    addrNodeTree first;
} nbTree;

/*** End Of User Defined Data Type ***/
/************************************/

/*****************************************************/
/***                 Operations                    ***/
/*****************************************************/

/*****************/
/** Constructor **/

//write the constructor here

void createTree(nbTree *tree);
/** Membuat tree kosong **/

void createInfo(infotype *X);
/** Membuat Isi dari infotype **/

/** End Of Constructor **/
/***********************/

/**************/
/** Accessor **/
```

```
/*** How to Access Root ?***/
/**
    Traversal Preorder (Root/Parent-Left-Right)
    Traversal PostOrder (Left-Right-Root/Parent)
    Traversal Inorder (Left-Root/Parent-Right)
Note :
    If Parent Have Children, Acces Children First
**/

//write code accessors here
void setInfo(addrNodeTree *A, infotype X);
/** Mengatur info dari alamat A sebagai info X **/

addrNodeTree getParentsAddr(nbTree tree, infotype X);
/** Mendapatkan address dari parents dengan infotype X **/

infotype getParentsInfo(nbTree tree, infotype x);
/**  Mendapatkan  infotype  parents  dengan  address  yang  diberikan  oleh  P
**/

infotype getChildInfo(nbTree tree, addrNodeTree C);
/** Mendapatkan infotype child dengan address yang diberikan oleh P **/

/** End of Accessor **/
/*******************/

/************/
/** Mutator **/

//write your code mutator
void addNextBrother(addrNodeTree *Root, infotype input);
/** Menambahkan node tree selain di node pertama pada **/
/** suatu level dengan alamat yang sudah dialokasi **/

void addFirst(addrNodeTree *Root, infotype input);
/** Menambahkan node tree di kiri (node pertama pada **/
/** suatu level) dengan alamat yang sudah dialokasi**/

void addRoot(nbTree *tree, infotype X);
/** Menambahkan root atau 3lement pertama dari tree **/

/** End Of Mutator **/
/*******************/
```

```
/***************/
/** Destructor **/

//write your destructor here
void delAll(nbTree *tree);
/** Semua tree akan dihapus dan di dealokasikan **/

void delByInfo(nbTree *tree, infotype X);
/** Menghapus element yang berisikan info X, bisa jadi
parent/root/child **/

/** End Of Destructor **/
/**********************/

/***************/
/** Read/Write **/

//write your code read/write
void printPreOrder(addrNodeTree Root);
/** Menampilkan node secara Pre Order : root, left, right **/

void printPostOrder(nbTree Root);
/** Menampilkan node secara Post Order : left, right, root **/

void printInOrder(nbTree Root);
/** Menampilkan node secara In Order : left, root , right **/

void printChildInfo(addrNodeTree parent);
/** Menampilkan anak-anak dari suatu parent**/

void printInfo(infotype X);
/** Menampilkan Info **/

/** End of Read/Write **/
/*********************/

/** additional **/
addrNodeTree Alokasi(infotype X);

void FindAddrByInfo(addrNodeTree root, infotype x, addrNodeTree *P);

bool CompareInfo(infotype A, infotype B);
```

```
void setName(infotype *x);

#endif
```

Pseudo Code Non Binary Tree

Struktur Data

```
// record yang menampung info pada setiap node tree
infotype : record{
                Nama       : array of character,
                Jabatan    : array of character
           }


Node : record {
           *addrNodeTree    : Node
           }

// alamat sebuah node tree
addrParent : addrNodeTree;

// record sebuah node tree
NodeTree : record {
                parent           : addrParent,
                info       : infotype,
                child      : addrNodeTree,
                next       : addrNodeTree
                }

// root pada sebuah tree
nbTree : record {
                first      : addrNodeTree
           }
```

Traversal

```
// print tree secara pre order : root, left, right
procedure printPreOrder(input/output Root : addrNodeTree){
      // Validasi apakah root ada
      if(Root != Null)
            print(Root→info)
      else
```

```
          print(“Root tidak ada”)

     // Rekursif terhadap anak pertama
     if(Root→child != Null)
          printPreOrder(Root→child)

     // Rekursif terhadap anak lainnya
     if(Root→next != Null)
          printPreOrder(Root→next)
}


// print tree secara post order : left, right, root
procedure printPostOrder(input/output Root : addrNodeTree){
     Pcur  : addrNodeTree
     Pcur ← Root

     // Rekursif terhadap anak pertama
     if(Pcur →child != Null)
          printPostOrder(Pcur→child)

     // tampilkan root
     print(Pcur→info)

     // Rekursif terhadap anak lainnya
     if(Pcur→next != Null)
          printPostOrder(Pcur→next)
}

// print tree secara level order : berdasarkan level/derajat
procedure printLevelOrder(input/output Root : addrNodeTree){
     Pcur  : addrNodeTree
     Q     : Queue
     n     : integer

     // Validasi root
     if(Root == Null)
          return

     enqueue(Root)
     While(Q.front != Null){
               // assignment ukuran Queue Q
               n ← size(Q)
               While(n > 0){
                    Pcur ← Q.front
                    print(Q.front→info)
                    dequeue(Q)
                    if(Pcur→child != Null{
                         repeat{
                              enqueue(Pcur)
                              Pcur ← Pcur→next
                         } until (Pcur != Null)
                    n--
               }
          }
     }
}
```

Implementasi dalam bentuk program bahasa C (File Body.c)

```c
#include "NonBinaryTree.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <Windows.h>

/******* Body ********/

void setInfo(nbTree *tree, infotype A, infotype B)
{
    /** Merubah node berinfo A menjadi B **/
    addrNodeTree P = NULL;
    FindAddrByInfo((*tree).first, A, &P);
    if (P != NULL)
    {
        P->info = B;
    }
}

addrNodeTree Alokasi(infotype X)
{
    /** Mengalokasikan memori untuk type X **/
    addrNodeTree P;
    P = (addrNodeTree)malloc(sizeof(NodeTree) + 1);
    if (P != NULL)
    {
        P->info = X;
        P->child = NULL;
        P->parents = NULL;
        P->next = NULL;
    }
    return P;
}

void createTree(nbTree *tree)
{
    /** Membuat tree kosong **/
    (*tree).first = NULL;
}

void addRoot(nbTree *tree, infotype X)
{
    /** Menambahkan root atau element pertama dari tree **/
    addrNodeTree P = Alokasi(X);
    if (P != NULL)
    {
        (*tree).first = P;
    }
    else
    {
        return;
    }
}

void createInfo(infotype *X)
{
    /** Membuat Isi dari infotype **/
    fflush(stdin);
    printf("Jabatan : ");
    scanf("%49[^\n]", (*X).Jabatan);
    fflush(stdin);
    printf("Nama    : ");
    scanf("%49[^\n]", (*X).Nama);
}

void addFirst(addrNodeTree *Root, infotype input)
{
    /** Menambahkan node tree di kiri (node pertama pada **/
    /** suatu level) dengan alamat yang sudah dialokasi**/
    addrNodeTree P = Alokasi(input);
    addrNodeTree curr;
    if (*Root != NULL && (*Root)->child == NULL)
    {
        curr = *Root;
        P->parents = curr;
        (*Root)->child = P;
    }
}

void addNextBrother(addrNodeTree *Root, infotype input)
{
    // addrNodeTree getParentsAddr(addrNodeTree root, infotype X)
    addrNodeTree P = Alokasi(input);
    if ((*Root) != NULL && P != NULL)
    {
        P->parents = (*Root)->parents;
        (*Root)->next = P;
    }
}

void printPreOrder(addrNodeTree Root)
{
    /** Menampilkan node secara Pre Order : root, left, right **/
    if (Root != NULL)
    {
        printInfo(Root->info);
        printf("\n");
    }
    else
    {
        printf("Sorry no one root available");
    }

    if (Root->child != NULL)
    {
        printPreOrder(Root->child);
    }

    if (Root->next != NULL)
    {
        printPreOrder(Root->next);
    }
}

void FindAddrByInfo(addrNodeTree root, infotype x, addrNodeTree *P)
{
    /** Guard if root = null **/
    if (root == NULL)
        return;

    //Search using preOrder transversal
    if (CompareInfo(root->info, x))
    {
        *P = root;
    }

    if (root->child != NULL)
    {
        FindAddrByInfo(root->child, x, &(*P));
    }

    if (root->next != NULL)
    {
        FindAddrByInfo(root->next, x, &(*P));
    }
}

infotype getParentsInfo(nbTree tree, infotype x)
{
     /** Mendapatkan infotype parents dengan address yang diberikan oleh
P **/
    infotype def;
    strcpy(def.Jabatan, "Not Find");
    strcpy(def.Nama, "Not Find");
    addrNodeTree P = NULL;
    FindAddrByInfo(tree.first, x, &P);
    if (P != NULL)
    {
        /* code */
        return P->parents->info;
    }
    else
    {
        return def;
    }
}

addrNodeTree getParentsAddr(nbTree tree, infotype x)
{
     /** Mendapatkan infotype parents dengan address yang diberikan oleh
P **/
    addrNodeTree P = NULL;
    FindAddrByInfo(tree.first, x, &P);
    if (P != NULL)
    {
        /* code */
        return P->parents;
    }
    else
    {
        return P;
    }
}

bool CompareInfo(infotype A, infotype B)
{
    /** Method untuk membadingkan Infotype **/
    if (strcmp(A.Nama, B.Nama) == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}

void setName(infotype *x)
{
    strcpy((*x).Jabatan, "default");
    //guard
    fflush(stdin);
    printf("Nama     : ");
    scanf("%49[^\n]", (*x).Nama);
}

void printInfo(infotype X)
{
    /** Menampilkan Info **/
    printf("Jabatan : %s\n", X.Jabatan);
    printf("Nama    : %s\n", X.Nama);
}
```

```c
void printChildInfo(addrNodeTree parent)
{
    /** Menampilkan anak-anak dari suatu parent**/
    if (parent != NULL)
    {
        printInfo(parent->info);
        printf("\n");
    }

    if (parent->next != NULL)
        printChildInfo(parent->next);
}
```

File Main.c

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <Windows.h>
#include <conio.h>
#include "NonBinaryTree.h"

int menu()
{
    int fitur;
    do
    {
        //clearkan layar monitor
        system("cls");
        //fitur menu
        printf("=== Program Organisasi ===\n");
        printf("1. Add Root\n");
        printf("2. Add Child\n");
        printf("3. Add Brother\n");
        printf("4. Print PreOrder\n");
        printf("5. Print PostOrder (Not Yet)\n");
        printf("6. Print InOrder (Not Yet)\n");
        printf("7. Print Lever Order (Not Yet)\n");
        printf("8. Info Atasan\n");
        printf("9. Info Bawahan\n");
        printf("0. Exit\n");
        printf("\nChoose : ");
        scanf("%d", &fitur);
    } while (fitur < 0 || fitur > 9);

    return fitur;
```

```
}

int main()
{
    //Deklarasi
    nbTree tree;
    infotype x, input;
    addrNodeTree current;
    int fitur = -1;

    //Inisialisasi
    current = NULL;
    createTree(&tree);

    //Make Apps Still a live untill user press exit
    while ((fitur = menu()) != 0)
    {
        switch (fitur)
        {
        case 1:
            //Input Info
            createInfo(&x);
            //masukan ke root
            addRoot(&tree, x);
            break;
        case 2:
            //print dulu tree
            printPreOrder(tree.first);
            //input dimana mau masuknya
            printf("Menambahakan bawahan siapa ?\n");
            setName(&x);
            //infotype untuk node baru
            printf("Siapakah yang ditambahkan ?\n");
            createInfo(&input);
            //cari alamatnya
            FindAddrByInfo(tree.first, x, &current);
            //masukan sebagai anak pertama dari current
            if (current != NULL)
                addFirst(&current, input);
            else
                printf("Sorry cant find that node");
            break;
        case 3:
            //print dulu tree
            printPreOrder(tree.first);
            //input dimana mau masuknya
            printf("Menambahakan satu level dimana ?\n");
            setName(&x);
            //infotype untuk node baru
            printf("Siapakah yang ditambahkan ?\n");
            createInfo(&input);
            //cari alamatnya
            FindAddrByInfo(tree.first, x, &current);
            //masukan sebagai brother dari current
            if (current != NULL)
                addNextBrother(&current, input);
            else
                printf("Sorry cant find that node\n");
            break;
        case 4:
            //pastikan layar bersih
            system("cls");
            printPreOrder(tree.first);
            break;
        case 8:
            printPreOrder(tree.first);
            printf("Siapa bawahannya ?\n");
            setName(&x);
                  printf("Atasannya adalah %s\n", getParentsInfo(tree,
x).Nama);
            break;
        case 9:
            printPreOrder(tree.first);
            printf("Siapa Atasannya ?\n");
            setName(&x);
            printf("Bawahannya adalah :\n");
            FindAddrByInfo(tree.first, x, &current);
            printChildInfo(current->child);
            break;
        default:
            printf("Mungkin fitur belum ada :)\n");
            break;
        }
        system("pause");
    }

    return 0;
}
```