

TUGAS JOBSHEET 06
PRAKTIKUM STRUKTUR DATA



DOSEN PENGAMPU:
Vera Irma Delianti, S.Pd., M.Pd.T.

OLEH:
M. Ilham
23343008

PROGRAM STUDI INFORMATIKA
DEPARTEMEN TEKNIK ELEKTRONIKA
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2024

1. binary-tree-1.c

a. SOURCE CODE

```
/* Nama file   : binary-tree-1.c
Pembuat       : M. ilham
Tgl pembuatan : 11 Maret 2024 */

#include <stdio.h>
#include <malloc.h>

struct nod {
    char data;
    struct nod *left;
    struct nod *right;
};

typedef struct nod NOD; // alias
typedef NOD POKOK;     // kok alias lagi ?

NOD *NodBaru(char item){ // okelah ini newnode
    NOD *n;
    n = (NOD *)malloc(sizeof(NOD));
    if(n != NULL){
        n->data = item;
        n->left = NULL;
        n->right = NULL;
    }
    return n;
}

void BinaryPokok(POKOK **T){ // fungsi utama
    *T = NULL;
}

typedef enum {
    FALSE = 0, TRUE = 1
} BOOL;
```

```

BOOL PokokKosong(POKOK *T) {
    return ((BOOL) (T == NULL));
}

void TambahNod(NOD **p, char item) {
    NOD *n;
    n = NodBaru(item);
    *p = n;
}

void preOrder(POKOK *T) {
    if(!PokokKosong(T)) {
        printf("%c", T->data);
        preOrder(T->left);
        preOrder(T->right);
    }
}

void inOrder(POKOK *T) {
    if(!PokokKosong(T)) {
        inOrder(T->left);
        printf("%c", T->data);
        inOrder(T->right);
    }
}

void postOrder(POKOK *T) {
    if(!PokokKosong(T)) {
        postOrder(T->left);
        postOrder(T->right);
        printf("%c", T->data);
    }
}

```

```

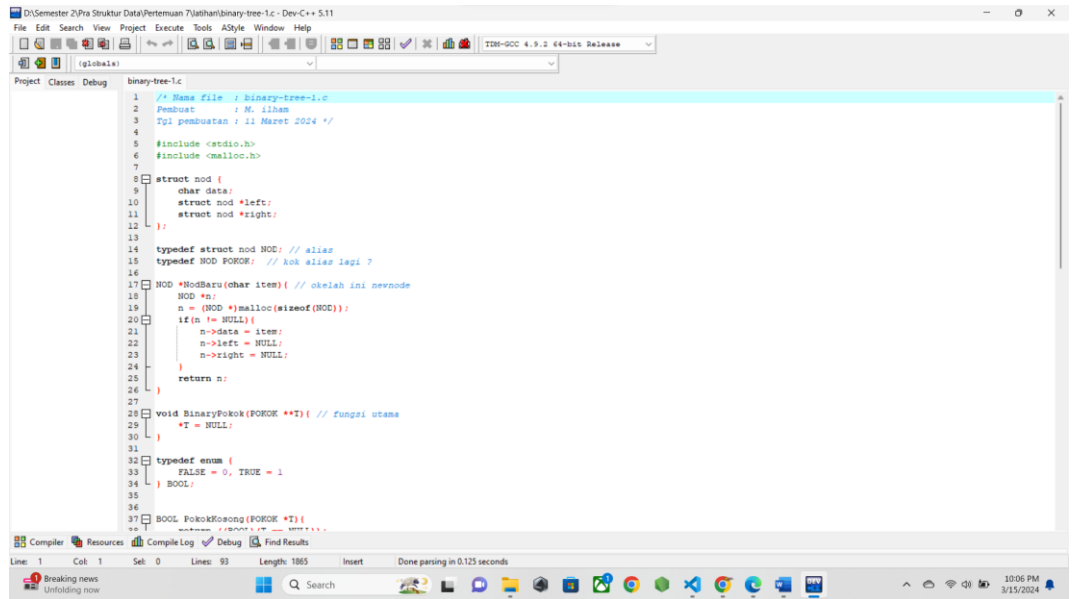
int main() {

    POKOK *kelapa;
    char buah;
    BinaryPokok(&kelapa);
    TambahNod(&kelapa, buah = 'M');
    TambahNod(&kelapa->left, buah = 'E');
    TambahNod(&kelapa->left->right, buah = 'I');
    TambahNod(&kelapa->right, buah = 'L');
    TambahNod(&kelapa->right->right, buah = 'O');
    TambahNod(&kelapa->right->right->left, buah =
'D');
    printf("Tampilan secara PreOrder: ");
    preOrder(kelapa);
    printf("\nTampilan secara InOrder: ");
    inOrder(kelapa);
    printf("\nTampilan secara PostOrder: ");
    postOrder(kelapa);

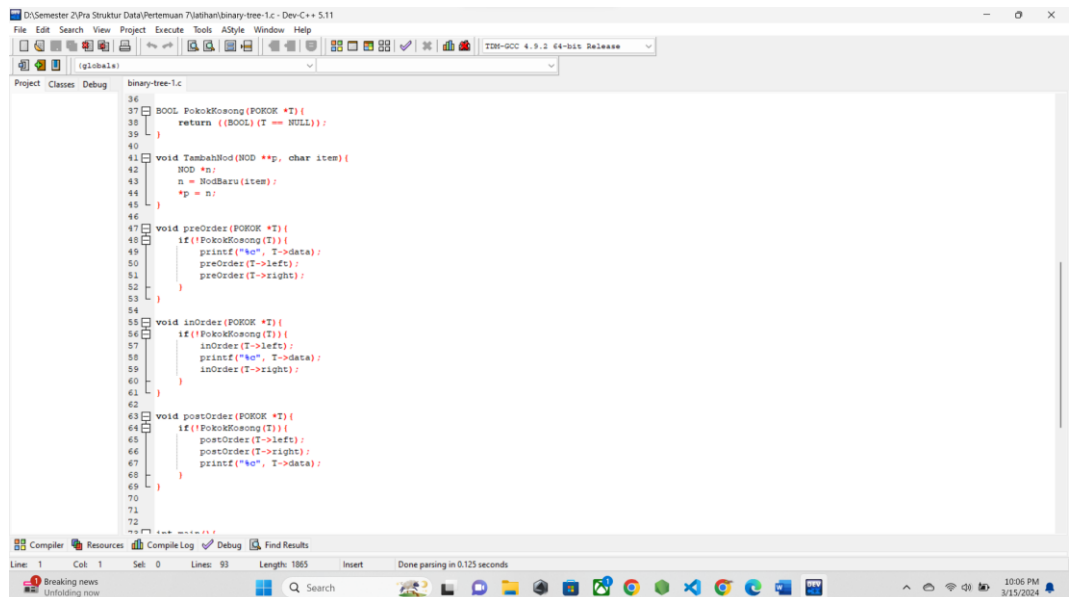
    return 0;
}

```

b. SCREENSHOT PROGRAM



```
1  // Nama file : binary-tree-1.c
2  PendosaP : M. Ilham
3  Tgl pembuatan : 11 Maret 2024 //
4
5  #include <stdio.h>
6  #include <malloc.h>
7
8  struct nod {
9      char data;
10     struct nod *left;
11     struct nod *right;
12 };
13
14 typedef struct nod NOD; // alias
15 typedef NOD POKOK; // kok alias lagi ?
16
17 NOD *ModBaru(char item) { // okelah ini newnode
18     NOD *n;
19     n = (NOD *)malloc(sizeof(NOD));
20     if (n != NULL) {
21         n->data = item;
22         n->left = NULL;
23         n->right = NULL;
24     }
25     return n;
26 }
27
28 void BinaryPokok(POKOK **T) // fungsi utama
29 *T = NULL;
30 }
31
32 typedef enum {
33     FALSE = 0, TRUE = 1
34 } BOOL;
35
36 BOOL PokokKosong(POKOK *T) {
```



```
36 BOOL PokokKosong(POKOK *T) {
37     return ((BOOL)(T == NULL));
38 }
39
40 void TambahNod(NOD **p, char item) {
41     NOD *n;
42     n = ModBaru(item);
43     *p = n;
44 }
45
46 void preOrder(POKOK *T) {
47     if (!PokokKosong(T)) {
48         printf("%c", T->data);
49         preOrder(T->left);
50         preOrder(T->right);
51     }
52 }
53
54 void inOrder(POKOK *T) {
55     if (!PokokKosong(T)) {
56         inOrder(T->left);
57         printf("%c", T->data);
58         inOrder(T->right);
59     }
60 }
61
62 void postOrder(POKOK *T) {
63     if (!PokokKosong(T)) {
64         postOrder(T->left);
65         postOrder(T->right);
66         printf("%c", T->data);
67     }
68 }
69
70
71
72
```

```

57     inOrder(T->left);
58     printf("%c", T->data);
59     inOrder(T->right);
60 }
61
62
63 void postOrder(POROK *T){
64     if(!PokokKosong(T)){
65         postOrder(T->left);
66         postOrder(T->right);
67         printf("%c", T->data);
68     }
69 }
70
71
72
73 int main(){
74
75     POROK *kelapa;
76     char buah;
77     BinaryPokok(kelapa);
78     TambahMod(kelapa, buah = 'M');
79     TambahMod(kelapa->left, buah = 'E');
80     TambahMod(kelapa->left->right, buah = 'I');
81     TambahMod(kelapa->right, buah = 'L');
82     TambahMod(kelapa->right->right, buah = 'O');
83     TambahMod(kelapa->right->right->left, buah = 'D');
84     printf("Tampilan secara PreOrder: ");
85     preOrder(kelapa);
86     printf("\nTampilan secara InOrder: ");
87     inOrder(kelapa);
88     printf("\nTampilan secara PostOrder: ");
89     postOrder(kelapa);
90
91     return 0;
92 }

```

c. SCREENSHOT OUTPUT

```

1  // Nama file : binary-tree-1.c
2  Pembuat   : M. ilhan
3  Tul pembuat : 11 Maret 2024 //
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  struct tnod {
9      char data;
10     struct tnod *left;
11     struct tnod *right;
12 }
13
14 typedef struct tnod type;
15
16 type *PokokKosong();
17 void BinaryPokok(type **);
18 void TambahMod(type **, char);
19 void preOrder(type *);
20 void inOrder(type *);
21 void postOrder(type *);
22
23 int main() {
24     type *kelapa;
25     char buah;
26     BinaryPokok(&kelapa);
27     TambahMod(kelapa, buah = 'M');
28     TambahMod(kelapa->left, buah = 'E');
29     TambahMod(kelapa->left->right, buah = 'I');
30     TambahMod(kelapa->right, buah = 'L');
31     TambahMod(kelapa->right->right, buah = 'O');
32     TambahMod(kelapa->right->right->left, buah = 'D');
33     printf("Tampilan secara PreOrder: ");
34     preOrder(kelapa);
35     printf("\nTampilan secara InOrder: ");
36     inOrder(kelapa);
37     printf("\nTampilan secara PostOrder: ");
38     postOrder(kelapa);
39     return 0;
40 }

```

Tampilan secara PreOrder: MEILOD
 Tampilan secara InOrder: EIMLOD
 Tampilan secara PostOrder: IEDOLM

 Process exited after 0.04318 seconds with return value 0
 Press any key to continue . . .

d. PENJELASAN PROGRAM

Pada program diatas adalah contoh implementasi binary tree. Pada program terdapat 7 buah fungsi buatan dan satu variabel enumeration yaitu BOOL dan juga 2 buah typedef, yaitu ada :

1. function struct pointer NodBaru
2. function void BinaryPokok
3. function BOOL PokokKosong
4. function void TambahNod
5. function void preorder

6. function void inOrder
7. function void postOrder
8. enumeration BOOL, dan
9. typedef struct nod Nod serta typedef struct NOD POKOK

Function NodBaru berfungsi untuk membuat node baru yang akan mengalokasikan memori pada heap dan menyimpan karakter yang dimasukkan lewat parameter ke dalam struct dengan member data dan menetapkan left dan rightnya NULL.

Function BinaryPokok berfungsi untuk memberikan alamat memori yang disimpan oleh variabel yang ditunjuk parameter bertipe variabel pointer struct nilai NULL.

Function PokokKosong berfungsi untuk mengecek apakah suatu variabel pointer struct menunjuk NULL, lalu akan di konversi dengan tipe data BOOL dan dikembalikan.

Function TambahNod berfungsi sebagai fungsi untuk menambahkan node baru, bedanya dengan NodBaru, fungsi ini merupakan outer function dari NodBaru, dimana fungsi ini memanggil fungsi NodBaru untuk mengalokasikan memori, lalu memori yang dialokasikan akan di kembalikan ke variabel pointer didalam fungsi TambahNod dan akan di simpan ke variabel yang ditunjuk oleh variabel pointer pada parameter fungsi.

Function preorder, inOrder, dan postOrder adalah function yang memiliki tugas hamper mirip yaitu untuk menampilkan data yang tersimpan dalam tree, tetapi ditampilkan dalam cara yang berbeda, dimana :

1. preOrder, menampilkan dengan cara : parent -> left -> right
 2. inOrder, menampilkan dengan cara : left -> parent -> right
 3. postOrder, menampilkan dengan cara : left -> right -> parent
- Enumeration digunakan untuk membuat tipe data bentukan berupa Boolean yang akan digunakan untuk menentukan true dan false

sehingga penulisannya lebih jelas karena tidak menggunakan tipe data int yang terkadang ambigu.

Typedef struct nod NOD digunakan untuk membuat alias dari struct nod menjadi NOD, lalu typedef NOD POKOK berfungsi untuk membuat alias dari struct nod yang di alias menjadi NOD sehingga menjadi POKOK.

Pada program saya, saya melakukan beberapa hal, yaitu membuat sebuah tree, lalu menambahkan 6 buah node baru masing masing yang menyimpan data karakter M, E, I, L, O, D.

Saat ditampilkan menggunakan pre order akan mendapatkan hasil : M
E I L O D

Saat ditampilkan menggunakan in order akan mendapatkan hasil :
EIMLDO

Saat ditampilkan menggunakan post order akan mendapatkan hasil :
IEDOLM

2. binary-tree-2.c

a. SOURCE CODE

```
/* Nama file   : binary-tree-2.c
Pembuat       : M. Ilham
Tgl pembuatan : 11 Maret 2024 */

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

struct data {
    int number;
    struct data *left, *right;
} *root;

void push(struct data **current, int number){
    if((*current) == NULL){
        (*current) = (struct data
*)malloc(sizeof(struct data));
        (*current)->number = number;
        (*current)->left = (*current)->right =
NULL;
    }
    else if(number < (*current)->number){
        push(&(*current)->left, number);
    }
    else if(number >= (*current)->number){
        push(&(*current)->right, number);
    }
}

void inOrder(struct data **current){
    if((*current) != NULL){
        inOrder(&(*current)->left);
        printf("%d -> ", (*current)->number);
        inOrder(&(*current)->right);
    }
}
```

```

}

void preOrder(struct data **current){
    if((*current) != NULL){
        printf("%d -> ", (*current)->number);
        preOrder(&(*current)->left);
        preOrder(&(*current)->right);
    }
}

void postOrder(struct data **current){
    if((*current) != NULL){
        postOrder(&(*current)->left);
        postOrder(&(*current)->right);
        printf("%d -> ", (*current)->number);
    }
}

void search(struct data **current, int number){
    if((*current) != NULL){
        if(number < (*current)->number){
            search(&(*current)->left, number);
        }
        else if(number > (*current)->number){
            search(&(*current)->right, number);
        }
        else {
            printf("Found : %d", (*current)-
>number);
        }
    }
    else {
        printf("Not Found.");
    }
}

```

```

int main() {

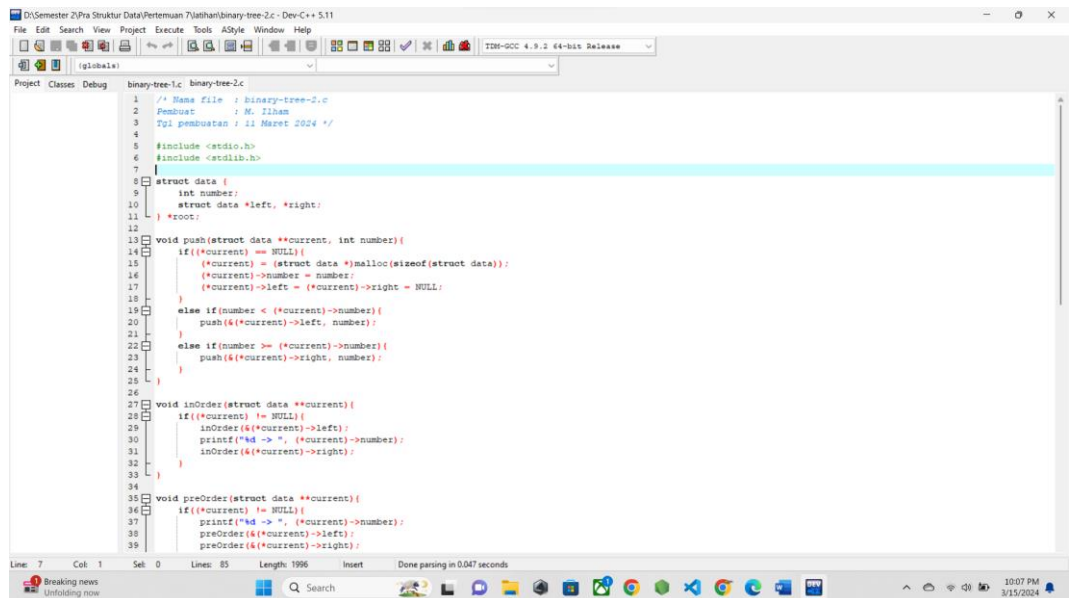
    push(&root, 11);
    push(&root, 22);
    push(&root, 13);
    push(&root, 15);
    push(&root, 9);
    inOrder(&root);
    printf("\n");
    preOrder(&root);
    printf("\n");
    postOrder(&root);
    printf("\n");
    search(&root, 91);
    getchar();

    return 0;

}

```

b. SCREENSHOT PROGRAM



```

D:\Semester 2\Pa Struktur Data\Perkenaan Tumbuhan\binary-tree-2.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globa1x)
binary-tree-1.c binary-tree-2.c
1 /* Name file : binary-tree-2.c
2 Pembuat : M. Ilham
3 Tgl pembuatan : 11 Maret 2024 */
4
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 struct data {
9     int number;
10    struct data *left, *right;
11 } *root;
12
13 void push(struct data **current, int number) {
14     if ((*current) == NULL) {
15         (*current) = (struct data *) malloc(sizeof(struct data));
16         (*current)->number = number;
17         (*current)->left = (*current)->right = NULL;
18     }
19     else if (number < (*current)->number) {
20         push(&(*current)->left, number);
21     }
22     else if (number >= (*current)->number) {
23         push(&(*current)->right, number);
24     }
25 }
26
27 void inOrder(struct data **current) {
28     if ((*current) != NULL) {
29         inOrder(&(*current)->left);
30         printf("%d -> ", (*current)->number);
31         inOrder(&(*current)->right);
32     }
33 }
34
35 void preOrder(struct data **current) {
36     if ((*current) != NULL) {
37         printf("%d -> ", (*current)->number);
38         preOrder(&(*current)->left);
39         preOrder(&(*current)->right);
40     }
41 }
42
43 int main() {
44     root = NULL;
45     push(&root, 11);
46     push(&root, 22);
47     push(&root, 13);
48     push(&root, 15);
49     push(&root, 9);
50     inOrder(&root);
51     printf("\n");
52     preOrder(&root);
53     printf("\n");
54     postOrder(&root);
55     printf("\n");
56     search(&root, 91);
57     getchar();
58
59     return 0;
60 }

```

```

47     printf("%d -> ", (*current)->number);
48 }
49 }
50
51 void search(struct data **current, int number){
52     if((*current) != NULL){
53         if(number < (*current)->number){
54             search(&(*current)->left, number);
55         }
56         else if(number > (*current)->number){
57             search(&(*current)->right, number);
58         }
59         else {
60             printf("Found : %d", (*current)->number);
61         }
62     }
63     else {
64         printf("Not Found.");
65     }
66 }
67
68 int main(){
69
70     push(&root, 11);
71     push(&root, 22);
72     push(&root, 13);
73     push(&root, 15);
74     push(&root, 9);
75     inOrder(&root);
76     printf("\n");
77     preOrder(&root);
78     printf("\n");
79     postOrder(&root);
80     printf("\n");
81     search(&root, 51);
82     getchar();
83
84     return 0;
85 }

```

c. SCREENSHOT OUTPUT

```

1 // Nama File : binary-tree-2.c
2 Pembuat : M. Ilham
3 Tgl pembuatan : 11 Maret 2024 //
4
5
6
7
8 9 -> 11 -> 13 -> 15 -> 22 ->
9 11 -> 9 -> 22 -> 13 -> 15 ->
9 9 -> 15 -> 13 -> 22 -> 11 ->
10 Not Found.
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

```

d. PENJELASAN PROGRAM

Program di atas merupakan implementasi dari binary tree. Pada program tersebut terdapat 5 buah fungsi yaitu fungsi push, preOrder, inOrder, postOrder, dan search.

Function push berfungsi untuk mengecek apakah alamat memori yang dimasukkan sebagai argument untuk parameter struct double pointer adakah null, jika iya maka program akan mengalokasikan memori di heap untuk variabel yang digunakan ebagai argument tadi, lalu data di

dalamnya akan di isi dengan menyalin inputan pada argument kedua dari fungsi dan menetapkan left dan right sebagai null.

Jika alamat yang dijadikan argument pada fungsi push bukan null, maka akan lanjut mengecek apakah parameter kedua yang diinputkan lebih kecil dari nilai yang tersimpan pada alamat di argument pertama. Jika iya maka akan melakukan rekursif dengan memanggil fungsi push lagi tetapi dengan argument pertamanya alamat kiri dari argument outer function.

Jika tidak maka akan mengecek lagi, apakah angka yang di inputkan sebagai argument kedua pada pemanggilan fungsi pertama lebih besar daripada nilai yang tersimpan pada alamat yang dijadikan argument. Jika iya, maka akan melakukan rekursif dengan memanggil fungsi push lagi tetapi dengan argument pertama alamat kanan dari argument di outer function.

Pada fungsi inOrder, preorder, dan postOrder memiliki fungsi yang hamper sama, perbedaannya hany pada bagaimana urutan mereka melakukan display, berikut deskripsi perbedaannya :

1. preOrder : parent -> left -> right
2. inOrder : left -> parent -> right
3. postOrder : left -> right -> parent

Tetapi perbedaannya dengan program pertama, disini yang menjadi parameter adalah variabel double pointer sehingga pada argumennya saat pemanggilan fungsi harus menggunakan alamat dari variabel pointer.

Function berikutnya adalah search yang memiliki parameter tree dan nilai yang akan di cari pada tree. Lalu akan di cek apakah struct tree yang ditunjuk bukan NULL. Jika bukan NULL, maka akan dicek lagi apakah angka yang disimpan parameter fungsi lebih kecil daripada angka yang disimpan oleh struct tree, jika iya maka akan dilakukan pemanggilan fungsi rekursif dengan argument pertama alamat kiri dari argument pada outer function.

Jika angka lebih besar, maka yang akan dipanggil adalah fungsi search dengan argument alamat kanan dari arrgumen pada outer function, dan jika bukan keduanya berarti angka yang dicari ketemu sehingga akan ditampilkan. Jika alamat yang disimpan pada argument fungsi search adalah NULL, maka akan menampilkan not found.

Pada program diatas saya menambahkan 5 buah data pada tree masing masing 11, 22, 13, 15, dan 9. Lalu saya tampilkan menggunakan inOrder preOrder, dan postOrder sehingga didapat hasil seperti yang tertera pada screnshoot. Lalu saya lakukan pencarian menggunakan fungsi search untuk mencari angka tertentu pada tree.

3. tugas-binary-tree-2.c

a. SOURCE CODE

```
/* Nama file   : tugas-binary-tree-2.c
Pembuat       : M. Ilham
Tgl pembuatan : 15 Maret 2024 */

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

struct Node {
    char data;
    struct Node *left, *right;
};

void createBinaryTree(struct Node **temp){
    *temp = NULL;
}

void addNode(struct Node **temp, char value){
    *temp = (struct Node *)malloc(sizeof(struct
Node));
    (*temp)->data = value;
    (*temp)->left = (*temp)->right = NULL;
}

void preOrder(struct Node **target){
    if((*target) != NULL){
        printf("%c -> ", (*target)->data);
        preOrder(&(*target)->left);
        preOrder(&(*target)->right);
    }
}

void inOrder(struct Node **target){
    if((*target) != NULL){
```

```

        inOrder(&(*target)->left);
        printf("%c -> ", (*target)->data);
        inOrder(&(*target)->right);
    }
}

void postOrder(struct Node **target){
    if((*target) != NULL){
        postOrder(&(*target)->left);
        postOrder(&(*target)->right);
        printf("%c -> ", (*target)->data);
    }
}

int main(){

    struct Node *root;
    createBinaryTree(&root);

    addNode(&root, 'R');
    addNode(&root->left, 'A');
    addNode(&root->left->left, 'S');
    addNode(&root->left->left->left, 'I');
    addNode(&root->left->left->right, 'T');
    addNode(&root->right, 'E');

    printf("Pre Order : ");
    preOrder(&root);
    printf("\n");
    printf("In Order : ");
    inOrder(&root);
    printf("\n");
    printf("Post Order : ");
    postOrder(&root);
    printf("\n");
}

```



```

        return 0;
    }
}

```

b. SCREENSHOT PROGRAM

```

D:\Semester 2\Prak Struktur Data\Pertemuan 7\tugas\tugas-binary-tree.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
[Icons] [Compiler] [Debugger] [Build] [Run] [Debug] [Tools] [Help] [TDM-GCC 4.9.2 64-bit Release]
Project Classes Debug
binary-tree-1.c binary-tree-2.c tugas-binary-tree.c
1  /* Nama File : tugas-binary-tree-2.c
2  Pembuat : M. Ilham
3  Tgl pembuatan : 14 Maret 2024 */
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  struct Node {
9      char data;
10     struct Node *left, *right;
11 };
12
13 void createBinaryTree(struct Node **temp){
14     *temp = NULL;
15 }
16
17 void addNode(struct Node **temp, char value){
18     *temp = (struct Node *)malloc(sizeof(struct Node));
19     (*temp)->data = value;
20     (*temp)->left = (*temp)->right = NULL;
21 }
22
23 void preOrder(struct Node **target){
24     if((*target) != NULL){
25         printf("%c -> ", (*target)->data);
26         preOrder(&(*target)->left);
27         preOrder(&(*target)->right);
28     }
29 }
30
31 void inOrder(struct Node **target){
32     if((*target) != NULL){
33         inOrder(&(*target)->left);
34         printf("%c -> ", (*target)->data);
35         inOrder(&(*target)->right);
36     }
37 }
38
39
40 void postOrder(struct Node **target){
41     if((*target) != NULL){
42         postOrder(&(*target)->left);
43         postOrder(&(*target)->right);
44         printf("%c -> ", (*target)->data);
45     }
46 }
47
48 int main(){
49
50     struct Node *root;
51     createBinaryTree(&root);
52
53     addNode(&root, 'R');
54     addNode(&root->left, 'A');
55     addNode(&root->left->left, 'S');
56     addNode(&root->left->left->left, 'I');
57     addNode(&root->left->left->right, 'T');
58     addNode(&root->right, 'E');
59
60     printf("Pre Order : ");
61     preOrder(&root);
62     printf("\n");
63     printf("In Order : ");
64     inOrder(&root);
65     printf("\n");
66     printf("Post Order : ");
67     postOrder(&root);
68     printf("\n");
69
70     return 0;
71 }
72
Line: 1 Col: 23 Sel: 0 Lines: 72 Length: 1582 Insert Done parsing in 0.015 seconds
72°F Mostly cloudy
Search
10:09 PM 3/15/2024

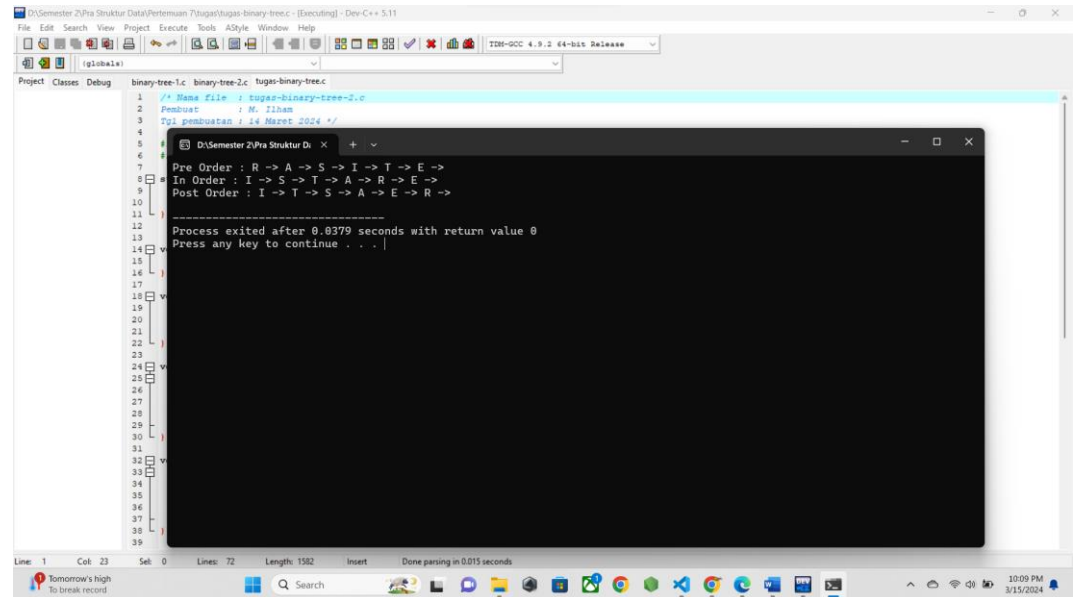
```

```

D:\Semester 2\Prak Struktur Data\Pertemuan 7\tugas\tugas-binary-tree.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
[Icons] [Compiler] [Debugger] [Build] [Run] [Debug] [Tools] [Help] [TDM-GCC 4.9.2 64-bit Release]
Project Classes Debug
binary-tree-1.c binary-tree-2.c tugas-binary-tree.c
34     inOrder(&(*target)->left);
35     printf("%c -> ", (*target)->data);
36     inOrder(&(*target)->right);
37 }
38
39
40 void postOrder(struct Node **target){
41     if((*target) != NULL){
42         postOrder(&(*target)->left);
43         postOrder(&(*target)->right);
44         printf("%c -> ", (*target)->data);
45     }
46 }
47
48 int main(){
49
50     struct Node *root;
51     createBinaryTree(&root);
52
53     addNode(&root, 'R');
54     addNode(&root->left, 'A');
55     addNode(&root->left->left, 'S');
56     addNode(&root->left->left->left, 'I');
57     addNode(&root->left->left->right, 'T');
58     addNode(&root->right, 'E');
59
60     printf("Pre Order : ");
61     preOrder(&root);
62     printf("\n");
63     printf("In Order : ");
64     inOrder(&root);
65     printf("\n");
66     printf("Post Order : ");
67     postOrder(&root);
68     printf("\n");
69
70     return 0;
71 }
72
Line: 1 Col: 23 Sel: 0 Lines: 72 Length: 1582 Insert Done parsing in 0.015 seconds
Tomorrow's high To break record
Search
10:09 PM 3/15/2024

```

c. SCREENSHOT OUTPUT



```
1  /* Name file : tugas-binary-tree-2.c
2  Pendapat : M. Izzah
3  Tol. pembuat : id. Nazet 2024 */
4
5
6
7  Pre Order : R -> A -> S -> I -> T -> E ->
8  In Order : I -> S -> T -> A -> R -> E ->
9  Post Order : I -> T -> S -> A -> E -> R ->
10
11 -----
12 Process exited after 0.0379 seconds with return value 0
13 Press any key to continue . . .
```

d. PENJELASAN PROGRAM

Program di atas adalah program untuk tugas yang merupakan implementasi untuk binary tree. Pada program tersebut terdapat 5 buah fungsi yaitu createBinaryTree, addNode, preOrder, inOrder, dan postOrder.

Tugas dari fungsi order adalah untuk menampilkan data sesuai dari jenis ordernya. Fungsi addNode berfungsi untuk mengalokasikan memori untuk variabel pada tree dan mengisi data pada alamat yang dialokasikan tersebut. Sedangkan createBinaryTree hanya membuat sebuah tree diawal mendapatkan nilai NULL.

Pada program saya, karena pada tugas disuruh untuk membuat program yang akan menampilkan hasil tertentu dengan menggunakan masing masing order, detailnya hasil yang akan di capai adalah :

preOrder : R A S I T E

inOrder : I S T A R E

postOrder : I T S A E R

Untuk mendapat hasil seperti yang diharapkan saya menggunakan fungsi untuk membuat tree, lalu saya tambahkan node baru dengan beberapa perhitungan sehingga didapat hasil akhir sebagai berikut :

```

      R
    A   E
  S
I     T

```

Lalu saya melakukan preOrder, inOrder, dan postOrder sehingga mendapatkan hasil output sesuai dengan yang diminta yaitu :

preOrder : R A S I T E

inOrder : I S T A R E

postOrder : I T S A E R