Laporan Resmi

Algoritma Struktur Data

14.1 Tree

Nama: Gagas Amukti Nandaka

Kelas: 1 D4 Teknik Informatika B

NRP: 3120600032

PROGRAM STUDI TEKNIK INFORMATIKA

DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2021

C. PERCOBAAN

1. Buatlah fungsi `bentuk()` untuk membentuk sebuah node baru untuk data yang telah diinputkan (alokasi node yang akan disisipkan)

2. Buatlah fungsi `sisip()` yang akan mengimplementasikan algoritma penyisipan node pada pembentukan binary tree.

3. Buatlah definisi fungsi `menu_kunjungan()` yang memberikan return value berupa nomor pilihan metode, dan memiliki tampilan sbb :

```
Pilih Penelusuran Tree
1. Preorder
2. PostOrder
3. Inorder
Pilihan anda :
```

4. Buatlah definisi `preorder(), inorder() dan postorder()` untuk mengimplementasikan algoritma masing-masing kunjungan preorder, inorder, dan postorder.

5. Buatlah definisi pada program utama untuk mendapatkan tampilan sbb :

```
MEMBENTUK SEBUAH TREE
Ketikkan data/infonya :
Ada Data lagi ? (y/t):
```

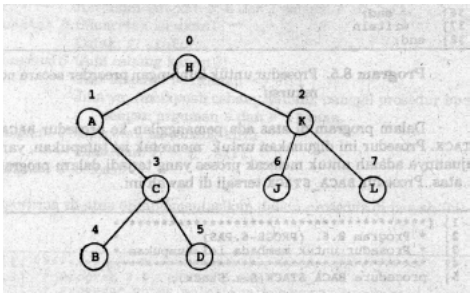
Hasil output berupa penelusuran linked list dari tree secara preorder, postorder dan inorder.

Misalnya dipilih menu preorder, maka tampilannya :

```
Hasil penelusuran menggunakan Preorder :
```

6. Buat fungsi `cari()` untuk mengimplementasikan proses pencarian data pada sebuah binary Tree

Gunaka n gambar tree berikut untuk melakukan uji coba
Input : H A K C J L B D



Listing Program :

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

struct pohon{
   char data;
   struct pohon *kiri;
   struct pohon *kanan;
};
struct pohon *root = NULL;
void masukdata(int data);
struct pohon* cari(int data);
void preorder(struct pohon* root);
void inorder(struct pohon* root);
void postorder(struct pohon* root);

int main(){
   int i, jum = 0, menu = 0;
   char carik;

   puts("PRAKTIKUM TREE SEARCHING GAGAS");
   printf("\nmau memasukkan berapa data ? : ");
   scanf("%d", &jum);
   char masuk[jum];
   fflush(stdin);

   for(i = 0; i < jum; i++){
    fflush(stdin);
    printf("masukkan data ke %d : ", i);
```

```
 scanf("%c", &masuk[i]);
 fflush(stdin);
}

for(i = 0; i < jum; i++)
   masukdata(masuk[i]);

do{
    puts("\nMenu TREE");
    puts("1.PreOrder Tree");
    puts("2.InOrder Tree");
    puts("3.PostOrder Tree");
    puts("4.ALL Tree");
    puts("5.Cari Tree");
    puts("6.Exit");
    printf("masukkan pilihan anda : ");
    scanf("%d", &menu);
    fflush(stdin);


    if(menu == 1){
     printf("Preorder Tree: ");
     preorder(root);
     printf("\n");
    }

    else if(menu == 2){
      printf("Inorder Tree: ");
      inorder(root);
      printf("\n");
```

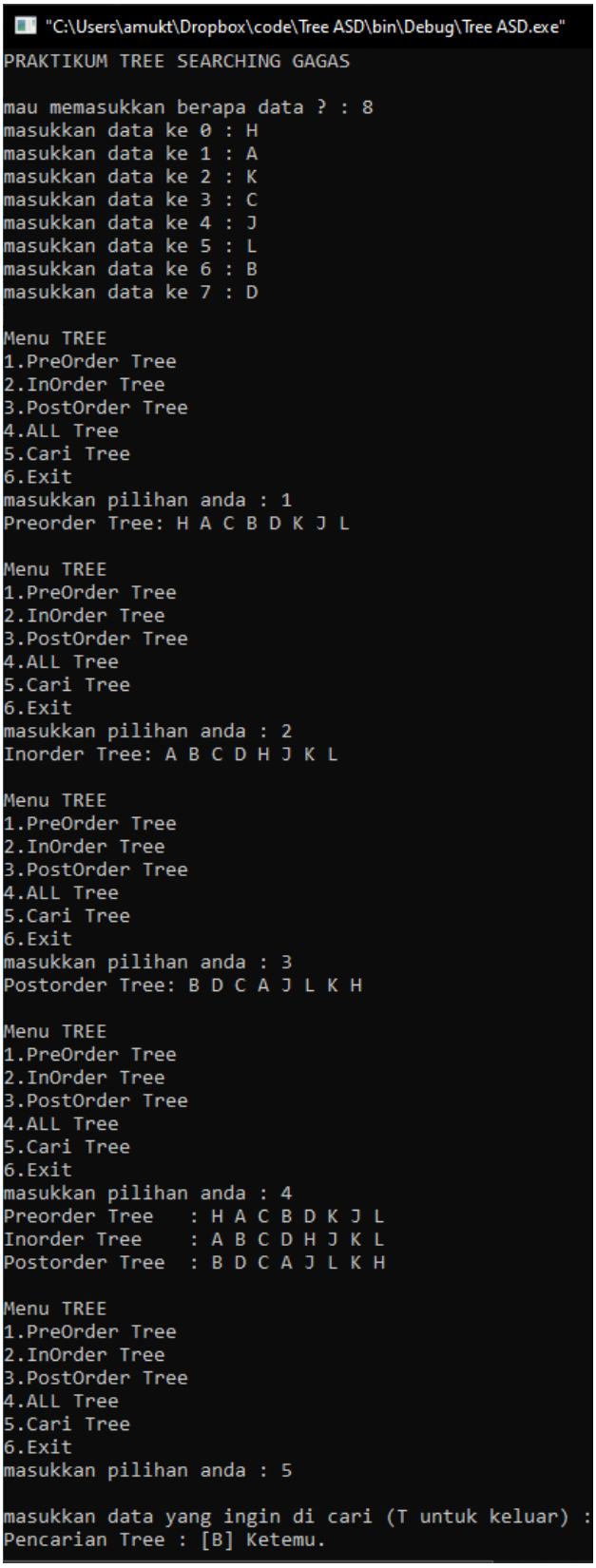
```
        }

        else if(menu == 3){
         printf("Postorder Tree: ");
         postorder(root);
         printf("\n");
        }

        else if(menu == 4){
         printf("Preorder Tree\t: ");
         preorder(root);

         printf("\nInorder Tree\t: ");
         inorder(root);

         printf("\nPostorder Tree\t: ");
         postorder(root);
         printf("\n");
        }

        else if(menu == 5){
            printf("\nmasukkan data yang ingin di cari (T untuk keluar)
: ");
            scanf("%c", &carik);
            fflush(stdin);

            struct pohon* temp = cari(carik);
            if(temp != NULL) {
               printf("[%c] Ketemu.", carik);
               printf("\n");
            }else{
```

```
            printf("[%c] Tidak ketemu :((.\n", carik);
          }
       }else{
        puts("Arigatu Gonzaimas");
        exit(0);
       }
   }while(1);
}

void masukdata(int data) {
   struct pohon *tempNode = (struct pohon*) malloc(sizeof(struct
pohon));
   struct pohon *current;
   struct pohon *parent;

   tempNode->data = data;
   tempNode->kiri = NULL;
   tempNode->kanan = NULL;

   if(root == NULL) {
      root = tempNode;
   }else {
      current = root;
      parent = NULL;

      while(1) {
         parent = current;

         if(data < parent->data) {
            current = current->kiri;
```

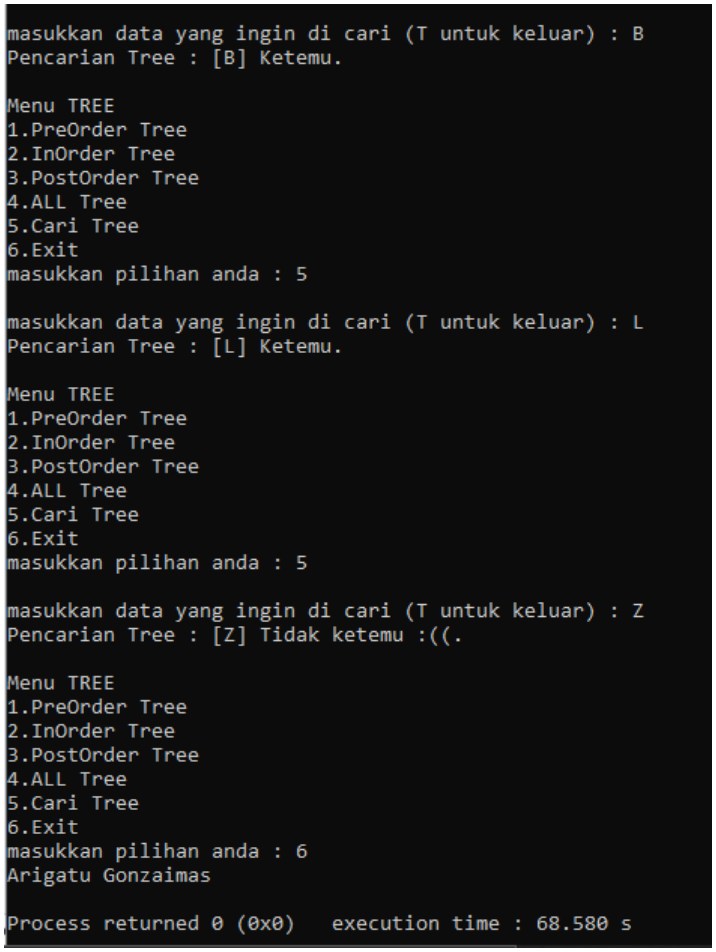
```
            if(current == NULL) {
               parent->kiri = tempNode;
               return;
            }
         }
         else {
            current = current->kanan;

            if(current == NULL) {
               parent->kanan = tempNode;
               return;
            }
         }
      }
   }
}

struct pohon* cari(int data) {
   struct pohon *current = root;
   printf("Pencarian Tree : ");

   while(current->data != data) {

      if(current->data > data) {
         current = current->kiri;
      }
      else {
         current = current->kanan;
      }
```

```
      if(current == NULL) {
         return NULL;
      }
   }
   return current;
}

void preorder(struct pohon* root) {
   if(root != NULL) {
      printf("%c ",root->data);
      preorder(root->kiri);
      preorder(root->kanan);
   }
}

void inorder(struct pohon* root) {
   if(root != NULL) {
      inorder(root->kiri);
      printf("%c ",root->data);
      inorder(root->kanan);
   }
}
void postorder(struct pohon* root) {
   if(root != NULL) {
      postorder(root->kiri);
      postorder(root->kanan);
      printf("%c ", root->data);
   }
}
```

Capture Output :



```
"C:\Users\amukt\Dropbox\code\Tree ASD\bin\Debug\Tree ASD.exe"
PRAKTIKUM TREE SEARCHING GAGAS

mau memasukkan berapa data ? : 8
masukkan data ke 0 : H
masukkan data ke 1 : A
masukkan data ke 2 : K
masukkan data ke 3 : C
masukkan data ke 4 : J
masukkan data ke 5 : L
masukkan data ke 6 : B
masukkan data ke 7 : D

Menu TREE
1.PreOrder Tree
2.InOrder Tree
3.PostOrder Tree
4.ALL Tree
5.Cari Tree
6.Exit
masukkan pilihan anda : 1
Preorder Tree: H A C B D K J L

Menu TREE
1.PreOrder Tree
2.InOrder Tree
3.PostOrder Tree
4.ALL Tree
5.Cari Tree
6.Exit
masukkan pilihan anda : 2
Inorder Tree: A B C D H J K L

Menu TREE
1.PreOrder Tree
2.InOrder Tree
3.PostOrder Tree
4.ALL Tree
5.Cari Tree
6.Exit
masukkan pilihan anda : 3
Postorder Tree: B D C A J L K H

Menu TREE
1.PreOrder Tree
2.InOrder Tree
3.PostOrder Tree
4.ALL Tree
5.Cari Tree
6.Exit
masukkan pilihan anda : 4
Preorder Tree    : H A C B D K J L
Inorder Tree     : A B C D H J K L
Postorder Tree   : B D C A J L K H

Menu TREE
1.PreOrder Tree
2.InOrder Tree
3.PostOrder Tree
4.ALL Tree
5.Cari Tree
6.Exit
masukkan pilihan anda : 5

masukkan data yang ingin di cari (T untuk keluar) :
Pencarian Tree : [B] Ketemu.
```



```
masukkan data yang ingin di cari (T untuk keluar) : B
Pencarian Tree : [B] Ketemu.

Menu TREE
1.PreOrder Tree
2.InOrder Tree
3.PostOrder Tree
4.ALL Tree
5.Cari Tree
6.Exit
masukkan pilihan anda : 5

masukkan data yang ingin di cari (T untuk keluar) : L
Pencarian Tree : [L] Ketemu.

Menu TREE
1.PreOrder Tree
2.InOrder Tree
3.PostOrder Tree
4.ALL Tree
5.Cari Tree
6.Exit
masukkan pilihan anda : 5

masukkan data yang ingin di cari (T untuk keluar) : Z
Pencarian Tree : [Z] Tidak ketemu :((.

Menu TREE
1.PreOrder Tree
2.InOrder Tree
3.PostOrder Tree
4.ALL Tree
5.Cari Tree
6.Exit
masukkan pilihan anda : 6
Arigatu Gonzaimas

Process returned 0 (0x0)   execution time : 68.580 s
```