LAPORAN PRAKTIKUM

STRUKTUR DATA DAN ALGORITMA

ANALISA SOURCE CODE PROGRAM TREE

DISUSUN OLEH

Muhammad Alwiza Ansyar M0520051

PROGRAM STUDI INFORMATIKA

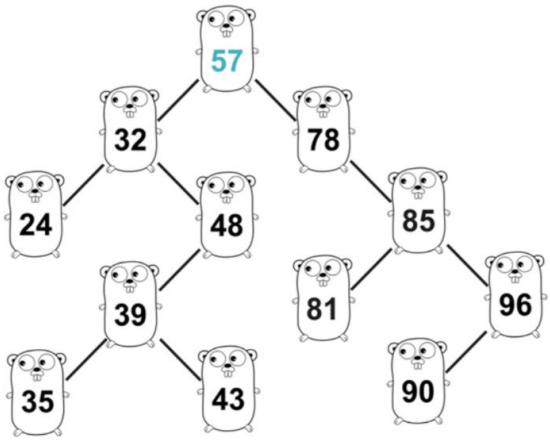
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SEBELAS MARET

2021

1. Langkah-langkah praktikum

- Buat program tree
- Masukkan data sesuai dari gambar berikut

Praktikum 3: Tree



- Lakukan tree traversal InOrder
- Buat laporan

2. Source code

PPSDA03_M0520051_Muhammad Alwiza Ansyar.cpp

```cpp
#include <iostream>
#include <array>
using namespace std;

typedef struct node{
    int data;
    node* Lptr;
    node* Rptr;
} Node;

void insert( Node** ptr, int value );
void inOrder_print( Node* ptr );

int main( void ){

    Node* root = NULL;

    array <int, 12> data = {57, 32, 78, 24, 48, 85, 39, 81, 96, 35, 43, 90};
    for( int i=0; i<12; i++ )
        insert( &root, data.at(i) );

    cout << "Data\t: ";
    for( int i=0; i<12; i++)
        cout << data.at(i) << " ";
    cout << endl;

    cout << "InOrder\t: ";
    inOrder_print( root );
}

void insert( Node** ptr, int value ){
    if( (*ptr)==NULL ){
        (*ptr) = new Node;
        (*ptr)->data = value;
        (*ptr)->Lptr = NULL;
        (*ptr)->Rptr = NULL;
    }
    else{
        if( value > (*ptr)->data ){
            insert( &( (*ptr)->Rptr ), value);
        }
        else if( value < (*ptr)->data ){
            insert( &( (*ptr)->Lptr ), value );
        }
    }
}

void inOrder_print( Node* ptr ){
    if( ptr!=NULL ){
        inOrder_print( ptr->Lptr );
        cout << ptr->data << " ";
        inOrder_print( ptr->Rptr );
    }

}
```

3. Output program

D:\Kuliah\Semester 2\Struktur Data & Algoritma\praksda\PPSDA03_M0520051_Muhammad Alwiza Ansyar.exe

```
Data    : 57 32 78 24 48 85 39 81 96 35 43 90
InOrder : 24 32 35 39 43 48 57 78 81 85 90 96
--------------------------------
Process exited after 0.04269 seconds with return value 0
Press any key to continue . . .
```

4. Analisa program

a.

```
1   #include <iostream>
2   #include <array>
3   using namespace std;
```

- Memuat library iostream dan array supaya dapat digunakan dalam program
- Mempersingkat penulisan fungsi dengan “using namespace std”

b.

```
5   typedef struct node{
6       int data;
7       node* Lptr;
8       node* Rptr;
9   } Node;
10
```

- Mendeklarasi struct dengan nama node yang berisi data bertipe integer dan dua *self referential* pointer bernama Lptr (L dari “Left”) dan Rptr (R dari “Right”), digunakan dalam membuat binary tree
- Pendeklarasian tadi sekaligus melakukan pengaliasan yaitu “typedef struct node Node”

c.

```
11  void insert( Node** ptr, int value );
12  void inOrder_print( Node* ptr );
13
```

- Mendeklarasi prototype dari fungsi insert dan inOrder_print

d.

```
14  int main( void ){
15
16      Node* root = NULL;
17
```

- Masuk ke main function
- Mendeklarasi pointer of node bernama root yang menunjuk ke NULL

e.

```
18      array <int, 12> data = {57, 32, 78, 24, 48, 85, 39, 81, 96, 35, 43, 90};
19      for( int i=0; i<12; i++ )
20          insert( &root, data.at(i) );
21
```

- Membuat array berisi data dari tree yang akan dimasukkan (data terdapat pada modul praktikum)
- Memasukkan data tadi ke dalam tree dengan cara melakukan for loop dengan setiap loop memanggil fungsi insert berargumen alamat root dan data sesuai index array (menggunakan fungsi array::at())

f.

```
22      cout << "Data\t: ";
23      for( int i=0; i<12; i++)
24          cout << data.at(i) << " ";
25      cout << endl;
26
27      cout << "InOrder\t: ";
28      inOrder_print( root );
```

- Menampilkan isi dari data sebagai output ke layar dengan cara melakukan for loop dan menampilkan data sesuai index array

- Menampilkan data dengan tree traversal tipe in order dengan cara memanggil fungsi inOrder_print berargumen variabel root

g.

```
31  void insert( Node** ptr, int value ){
32      if( (*ptr)==NULL ){
33          (*ptr) = new Node;
34          (*ptr)->data = value;
35          (*ptr)->Lptr = NULL;
36          (*ptr)->Rptr = NULL;
37      }
38      else{
39          if( value > (*ptr)->data ){
40              insert( &( (*ptr)->Rptr ), value);
41          }
42          else if( value < (*ptr)->data ){
43              insert( &( (*ptr)->Lptr ), value );
44          }
45      }
46  }
47
```

- Fungsi insert dengan return type void dan parameter Node** bernama ptr dan integer bernama value. Digunakan double pointer karena untuk memodifikasi variabel bertipe Node* maka digunakan pointer of Node* (yaitu Node**)
- Melakukan If else statement untuk menyeleksi lokasi penempatan data pada tree. Hanya akan dilakukan pendefinisian node baru apabila ptr menunjuk ke NULL
- Pada if, terdapat kondisi yaitu ptr menunjuk ke NULL, dilakukan alokasi memori dengan fungsi new (baris 33), memberikan nilai dari variabel value ke variabel data dari ptr (baris 34), dan mengatur supaya Lptr dan Rptr menunjuk ke NULL (baris 35-36)
- Pada else, berarti memiliki kondisi ptr menunjuk ke suatu node, dilakukan lagi if else statement untuk menyeleksi ke arah mana ptr akan melintas (kiri atau kanan). Pada if statement (baris 39) terdapat kondisi nilai value > data dari node yang ditunjuk ptr (*ptr->data), jika True maka dilakukan rekursi yaitu pemanggilan fungsi insert kembali dengan mengirim alamat Rptr dari node yang ditunjuk ptr (*ptr->Rptr) dan variabel value (baris 40). Sama halnya dengan else if, yang membedakan ialah kondisi nilai value < data dari node yang ditunjuk ptr (baris 42) dan ptr melintas ke kiri (baris 43)

h.

```
48  void inOrder_print( Node* ptr ){
49      if( ptr!=NULL ){
50          inOrder_print( ptr->Lptr );
51          cout << ptr->data << " ";
52          inOrder_print( ptr->Rptr );
53      }
54
55  }
```

- Fungsi inOrder_print dengan return type void dan parameter Node* bernama ptr
- Terdapat if statement dengan kondisi ptr tidak menunjuk ke NULL
- Dilakukan rekursi yaitu pemanggilan fungsi inOrder_print berargumen Lptr dari ptr (baris 50)
- Menampilkan data dari node yang ditunjuk ptr menggunakan cout
- Dilakukan rekursi yaitu pemanggilan fungsi inOrder_print berargumen Rptr dari ptr (baris 52)

5. Kesimpulan

- Tree pada program ini berjenis Binary Search Tree (BST) yaitu tree dengan children tidak lebih dari dua dan penyeleksian lokasi node tergantung dari value nya, apabila lebih besar dari root maka akan ditempatkan di kanan, sedangkan apabila lebih kecil dari root maka akan ditempatkan di kiri.
- InOrder memiliki pola (left, root, right) yang lalu diimplementasikan sebagai (rekursi fungsi dengan argument ptr->Lptr, menampilkan ptr->data, rekursi fungsi dengan argument ptr->Rptr)