TUGAS MATA KULIAH STRUKTUR DATA PERTEMUAN KE-7

Nama : Afridho Ikhsan
Kelas : 2A - Informatika
NPM : 2210631170002

Kodingan :

```
 5   #include <iostream>
 6   using namespace std;
 7   #define maks 20
 8
 9   struct node //Mendeklarasikan struct node untuk
     menampung data & pointer ke left child & right child
     dari masing-masing node pada tree
10   {
11       int data;
12       struct node *left;
13       struct node *right;
14   };
15
16   struct Queue //Mendeklarasikan Queue yang nanti akan
     digunakan untuk membangun & penelusuran tree
     berdasarkan level
17   {
18       int rear, front;
19       struct node *alamatNode[maks];
20   } nodeQueue;
21
22   bool isFull(Queue *selectedQueue) //Untuk mengecek
     apakah Queue yang menampung alamat dari node tree sudah
     penuh
23   {
24       return selectedQueue->rear == maks;
25   }
26
27   bool isEmpty(Queue *selectedQueue) //Untuk mengecek
     apakah Queue yang menampung alamat dari node tree masih
     kosong
28   {
29       return selectedQueue->rear == 0;
30   }
```

```
32  void enqueue(Queue *selectedQueue, struct node
    *enqueuedPointer) //Untuk memasukkan alamat dari node
    tree ke dalam queue
33  {
34      if (isFull(selectedQueue))
35      {
36          std::cout << "Antrian sudah penuh!" << endl;
37      }
38      else
39      {
40          selectedQueue->alamatNode[selectedQueue->rear]
            = enqueuedPointer;
41          selectedQueue->rear++;
42      }
43  }
44
45  struct node *dequeue(Queue *selectedQueue) //Untuk
    mengambil alamat dari node tree yang paling depan pada
    queue
46  {
47      struct node *temp = selectedQueue->alamatNode
        [selectedQueue->front];
48      for (int i = selectedQueue->front; i <
        selectedQueue->rear; i++)
49      {
50          selectedQueue->alamatNode[i] =
            selectedQueue->alamatNode[i + 1];
51      }
52      selectedQueue->rear--;
53      return temp;
54  }
55
```

```c
int cekBatasKedalaman(struct node **selectedTree) // Untuk melakukan penelusuran jalur yang paling pendek dari akar ke leaf, sehingga dapat diketahui batas dalamnya
{
    node *temp;

    enqueue(&nodeQueue, *selectedTree);
    int batasKedalaman = 1;
    while (!isEmpty(&nodeQueue))
    {
        temp = dequeue(&nodeQueue);
        if (temp->left == NULL || temp->right == NULL)
        {
            return batasKedalaman;
        }
        else
        {
            batasKedalaman++;
            if (temp->left != NULL)
            {
                enqueue(&nodeQueue, temp->left);
            }
            else if (temp->right != NULL)
            {
                enqueue(&nodeQueue, temp->right);
            }
        }
    }
    return batasKedalaman;
}
```

```
86  void membangunTree() //Untuk membangun tree berdasarkan
    konsep binary tree
87  {
88      node *p, *t, *root;
89      p = new node();
90      root = new node();
91
92      int x;
93      cout << "Inputkan value untuk root : ";
94      cin >> x;
95      root->data = x;
96      root->left = root->right = NULL;
97      enqueue(&nodeQueue, root);
98
99      while (!isEmpty(&nodeQueue))
100     {
101         int y, z;
102         p = dequeue(&nodeQueue);
103         cout << "Inputkan nilai dari left child dari "
            << p->data << " : ";
104         cin >> y;
105         if (y != -1)
106         {
107             t = new node();
108             t->data = y;
109             t->left = t->right = 0;
110             p->left = t;
111             enqueue(&nodeQueue, t);
112         }
```

```
114             cout << "Inputkan nilai dari right child dari "
                << p->data << " : ";
115             cin >> z;
116             if (z != -1)
117             {
118                 t = new node();
119                 t->data = z;
120                 t->left = t->right = 0;
121                 p->right = t;
122                 enqueue(&nodeQueue, t);
123             }
124         }
125         int testdata = cekBatasKedalaman(&root);
126         cout << "Batas kedalaman : " << testdata << endl;
127     }
128
129     int main()
130     {
131         membangunTree();
132     }
```

Kodingan :

Proses pembangunan tree dilakukan berdasarkan urutan level by level :

```
Inputkan value untuk root : 3
Inputkan nilai dari left child dari 3 : 9
Inputkan nilai dari right child dari 3 : 20
Inputkan nilai dari left child dari 9 : -1
Inputkan nilai dari right child dari 9 : -1
Inputkan nilai dari left child dari 20 : 15
Inputkan nilai dari right child dari 20 : 7
Inputkan nilai dari left child dari 15 : -1
Inputkan nilai dari right child dari 15 : -1
Inputkan nilai dari left child dari 7 : -1
Inputkan nilai dari right child dari 7 : -1
Batas kedalaman : 2
```

Child node yang diberi input bernilai -1 menandakan ketidakmilikan child/degree dari suatu node

Misal pada baris ke 4, " Inputkan nilai dari left child dari 9 : -1", artinya node yang bernilai 9 tersebut tidak memiliki left child (left child dari node tersebut = NULL)

Jika direpresentasikan dengan gambar, tree yang dibangun pada program di atas menjadi seperti yang telah diperintahkan pada tugas :