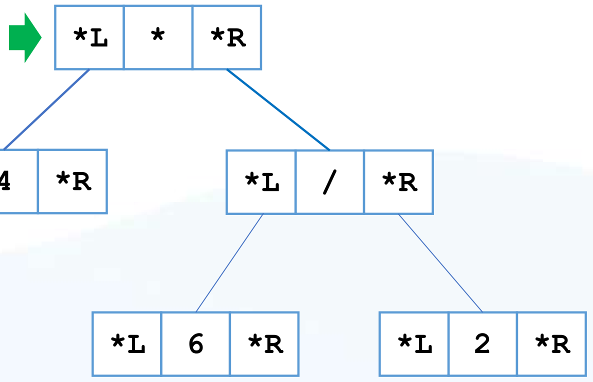
PROGRAM STUDI
TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO

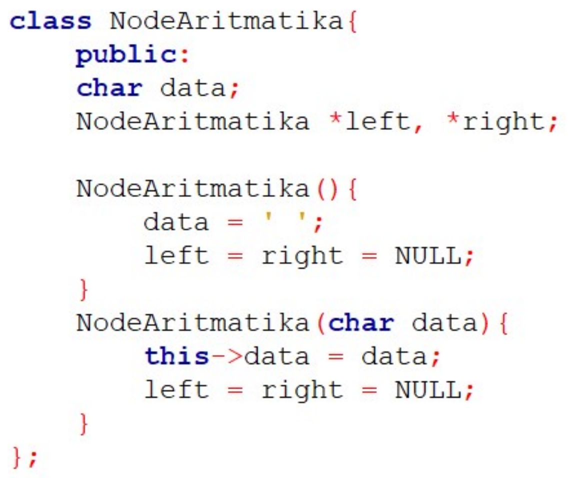
Mata Kuliah
Algoritma dan
Struktur Data



Tree Aritmatika

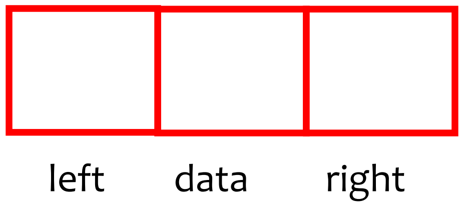
Tim Pengampu Mata Kuliah Algoritma dan Struktur Data

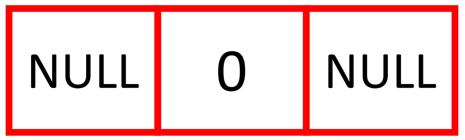
Class NodeAritmatika



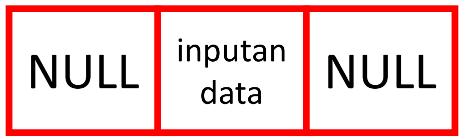
```
class NodeAritmatika{
    public:
    char data;
    NodeAritmatika *left, *right;

    NodeAritmatika(){
        data = ' ';
        left = right = NULL;
    }
    NodeAritmatika(char data){
        this->data = data;
        left = right = NULL;
    }
};
```

pointer right berisi alamat Node selanjutnya
pointer left berisi alamat Node sebelumnya





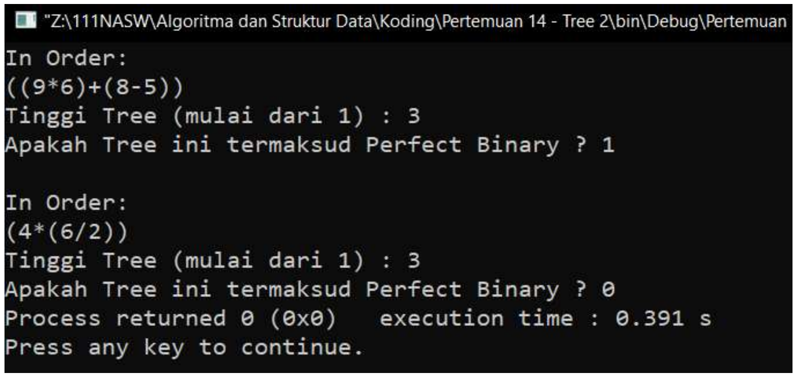
Konstruktor default



Konstruktor inputan
Data tergantung inputan

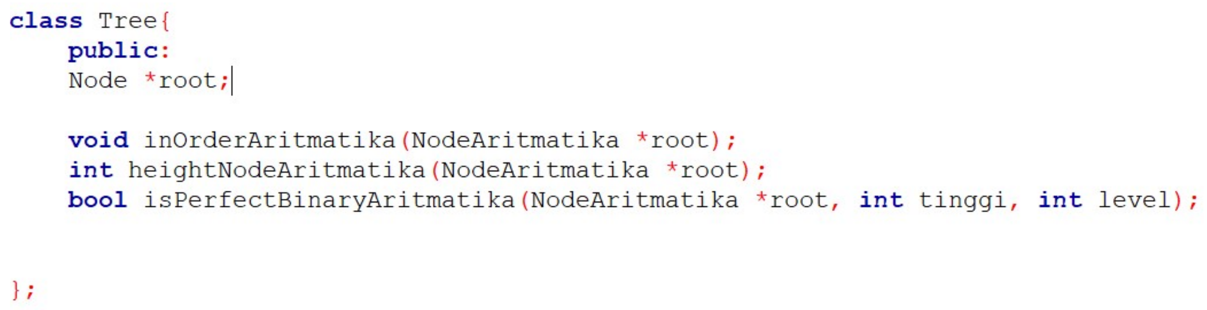
int main

```
/* representasi tree
        +
      /   \
     *     -
   / \    / \
  9   6  8   5  */
NodeAritmatika *root = new NodeAritmatika('+');
root->left = new NodeAritmatika('*');  root->right = new NodeAritmatika('-');
root->left->left = new NodeAritmatika('9'); root->left->right = new NodeAritmatika('6');
root->right->left = new NodeAritmatika('8'); root->right->right = new NodeAritmatika('5');
Tree T;
int tinggi;

cout << "In Order: " << endl; T.inOrderAritmatika(root);

tinggi = T.heightNodeAritmatika(root);
cout <<"\nTinggi Tree (mulai dari 1) : " << tinggi;

cout <<"\nApakah Tree ini termaksud Perfect Binary ? " <<
        T.isPerfectBinaryAritmatika(root, tinggi, 0);

/* representasi tree
        *
      /   \
     4     /
          / \
         6   2 */
NodeAritmatika *akar = new NodeAritmatika('*');
akar->left = new NodeAritmatika('4');  akar->right = new NodeAritmatika('/');
akar->right->left = new NodeAritmatika('6'); akar->right->right = new NodeAritmatika('2');
Tree pohon;
int tinggiPohon;

cout << "\n\nIn Order: " << endl; pohon.inOrderAritmatika(akar);

tinggiPohon = pohon.heightNodeAritmatika(akar);
cout <<"\nTinggi Tree (mulai dari 1) : " << tinggiPohon;

cout <<"\nApakah Tree ini termaksud Perfect Binary ? " << pohon.isPerfectBinaryAritmatika(akar, tinggiPohon, 0);
```



```
"Z:\111NASW\Algoritma dan Struktur Data\Koding\Pertemuan 14 - Tree 2\bin\Debug\Pertemuan
In Order:
((9*6)+(8-5))
Tinggi Tree (mulai dari 1) : 3
Apakah Tree ini termaksud Perfect Binary ? 1

In Order:
(4*(6/2))
Tinggi Tree (mulai dari 1) : 3
Apakah Tree ini termaksud Perfect Binary ? 0
Process returned 0 (0x0)   execution time : 0.391 s
Press any key to continue.
```

Class Tree



```
class Tree{
    public:
    Node *root;

    void inOrderAritmatika(NodeAritmatika *root);
    int heightNodeAritmatika(NodeAritmatika *root);
    bool isPerfectBinaryAritmatika(NodeAritmatika *root, int tinggi, int level);


};
```

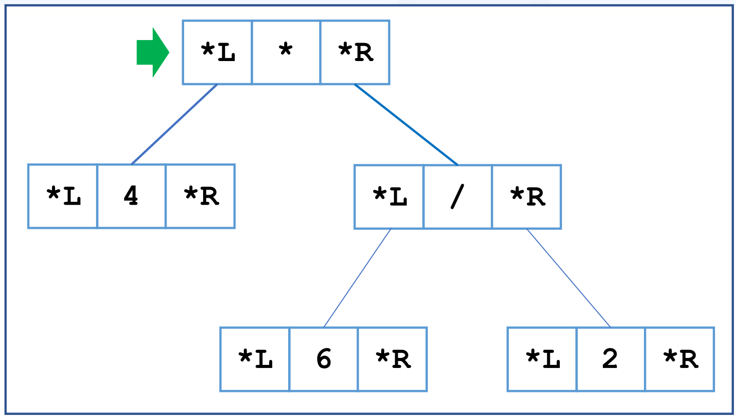
FAKULTAS ILMU KOMPUTER

inOrderAritmatika(root) #1

```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
        inOrderAritmatika(root->right);
        if (root->left != NULL || root->right != NULL)
            cout << ")";
    }
}
```

Cetak = (

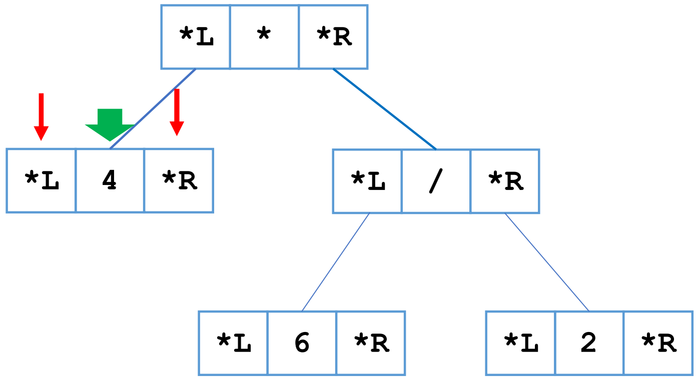
Cetak = (4*



```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
        inOrderAritmatika(root->right);
        if (root->left != NULL || root->right != NULL)
            cout << ")";
    }
}
```

Cetak = (4

```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
        inOrderAritmatika(root->right);
        if (root->left != NULL || root->right != NULL)
            cout << ")";
    }
}
```



```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
```

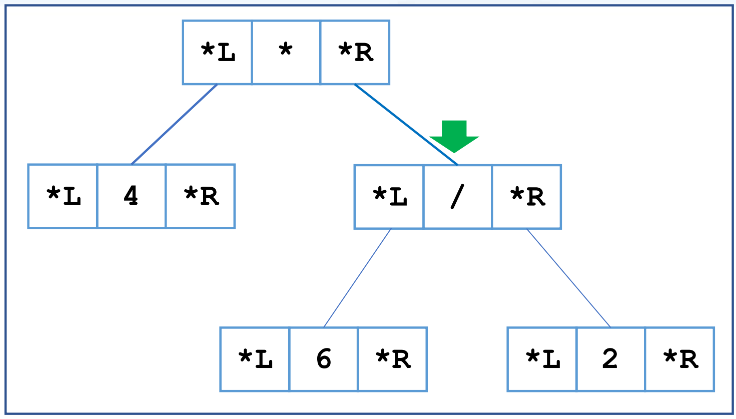
FAKULTAS ILMU KOMPUTER

inOrderAritmatika(root) #2

```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
        inOrderAritmatika(root->right);
        if (root->left != NULL || root->right != NULL)
            cout << ")";
    }
}
```

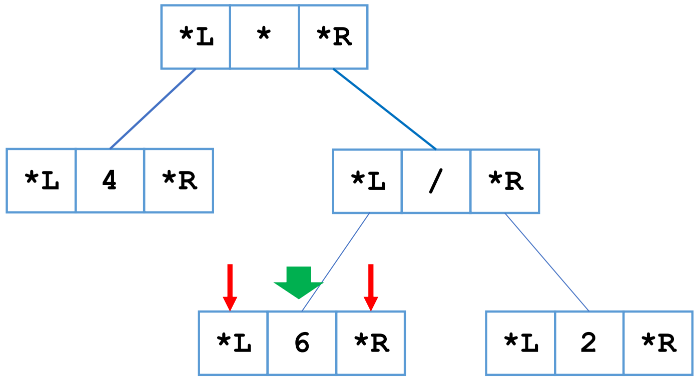
Cetak = (4*(

Cetak = (4*(6/



```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
        inOrderAritmatika(root->right);
        if (root->left != NULL || root->right != NULL)
            cout << ")";
    }
}
```

Cetak = (4*(6



```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
        inOrderAritmatika(root->right);
        if (root->left != NULL || root->right != NULL)
            cout << ")";
    }
}
```

```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
```

FAKULTAS ILMU KOMPUTER

inOrderAritmatika(root) #3

```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
        inOrderAritmatika(root->right);
        if (root->left != NULL || root->right != NULL)
            cout << ")";
    }
}
```

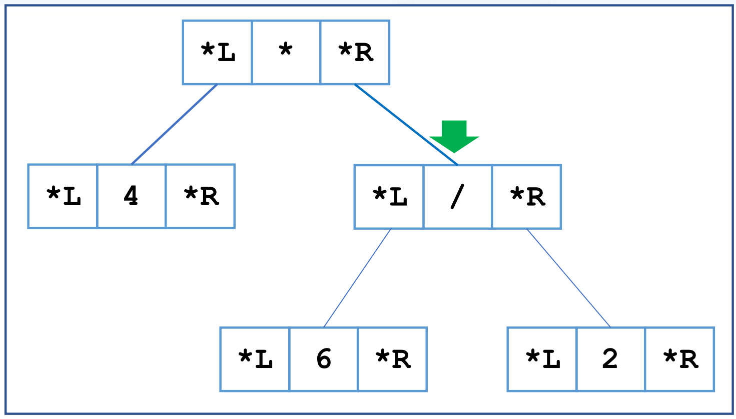
Cetak = (4*(6/2)

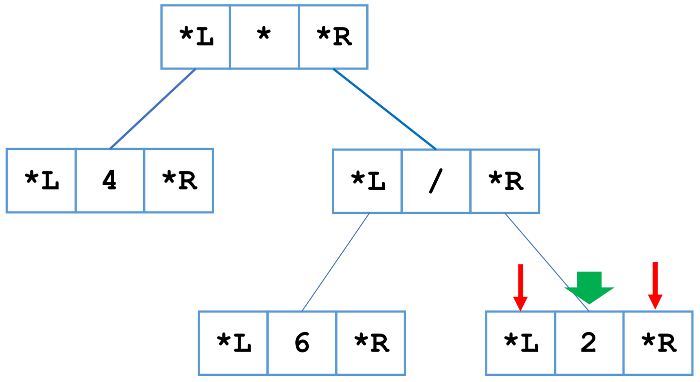
```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
        inOrderAritmatika(root->right);
        if (root->left != NULL || root->right != NULL)
            cout << ")";
    }
}
```

Cetak = (4*(6/2

```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
        inOrderAritmatika(root->right);
        if (root->left != NULL || root->right != NULL)
            cout << ")";
    }
}
```

```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
```

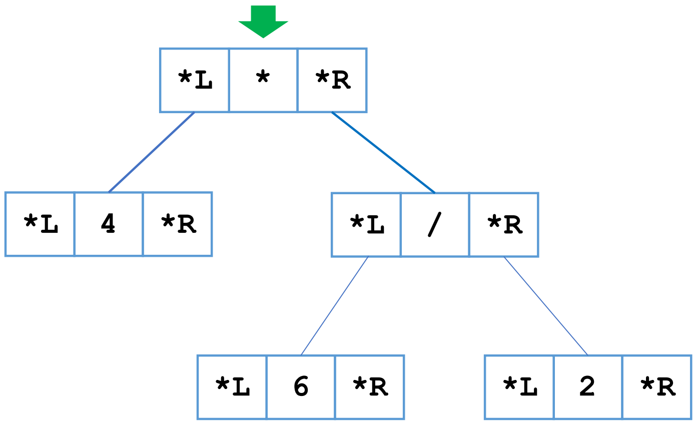




inOrderAritmatika(root) #4

```
void Tree::inOrderAritmatika(NodeAritmatika *root){
    if(root != NULL){
        if (root->left != NULL || root->right != NULL)
            cout << "(";
        inOrderAritmatika(root->left);
        cout << root->data;
        inOrderAritmatika(root->right);
        if (root->left != NULL || root->right != NULL)
            cout << ")";
    }
}
```

Cetak = (4*(6/2))



heightNodeAritmatika

```
int Tree::heightNodeAritmatika(NodeAritmatika *root){
    if(root == NULL){
        return 0;
    }else{
        int leftH=leftH=heightNodeAritmatika(root->left);
        int rightH= leftH=heightNodeAritmatika(root->right);

        return 1+max(leftH, rightH);
    }
}
```

Sama seperti heightNode

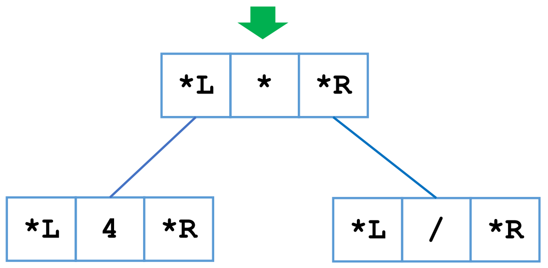
isPerfectBinaryAritmatika(root, 2, 0) output true #1

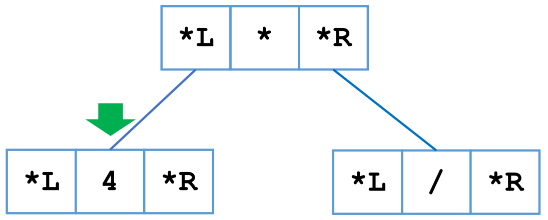
```
bool Tree::isPerfectBinaryAritmatika(NodeAritmatika *root, int tinggi, int level){
    if (root == NULL)
        return true;

    if (root->left == NULL && root->right == NULL)
        return (tinggi == level+1);

    if (root->left == NULL || root->right == NULL)
        return false;

    return isPerfectBinaryAritmatika(root->left, tinggi, level+1) &&
           isPerfectBinaryAritmatika(root->right, tinggi, level+1);
}
```

true &&

isPerfectBinaryAritmatika(root->left, 2, 1)

```
bool Tree::isPerfectBinaryAritmatika(NodeAritmatika *root, int tinggi, int level){
    if (root == NULL)
        return true;

    if (root->left == NULL && root->right == NULL)
        return (tinggi == level+1);   return (2==1); return true

    if (root->left == NULL || root->right == NULL)
        return false;

    return isPerfectBinaryAritmatika(root->left, tinggi, level+1) &&
           isPerfectBinaryAritmatika(root->right, tinggi, level+1);
}
```





isPerfectBinaryAritmatika(root, 2, 0) output true #2

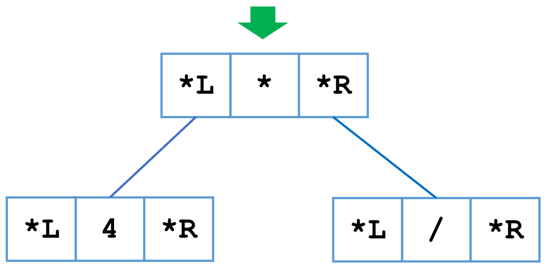
```
bool Tree::isPerfectBinaryAritmatika(NodeAritmatika *root, int tinggi, int level){
    if (root == NULL)
        return true;

    if (root->left == NULL && root->right == NULL)
        return (tinggi == level+1);

    if (root->left == NULL || root->right == NULL)
        return false;

    return isPerfectBinaryAritmatika(root->left, tinggi, level+1) &&
           isPerfectBinaryAritmatika(root->right, tinggi, level+1);
}
```

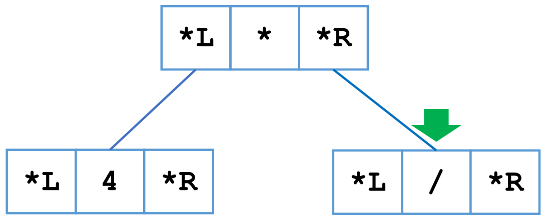
true && true

return true

isPerfectBinaryAritmatika(root->right, 2, 1)

```
bool Tree::isPerfectBinaryAritmatika(NodeAritmatika *root, int tinggi, int level){
    if (root == NULL)
        return true;

    if (root->left == NULL && root->right == NULL)
        return (tinggi == level+1);  return (2==1); return true

    if (root->left == NULL || root->right == NULL)
        return false;

    return isPerfectBinaryAritmatika(root->left, tinggi, level+1) &&
           isPerfectBinaryAritmatika(root->right, tinggi, level+1);
}
```





isPerfectBinaryAritmatika(root, 2, 0) output false #1

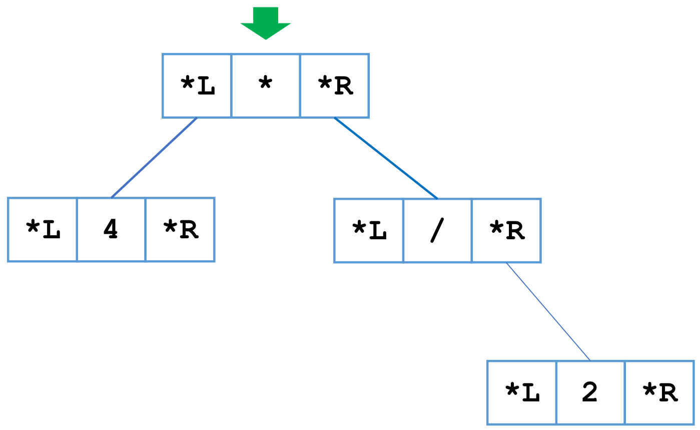
```
bool Tree::isPerfectBinaryAritmatika(NodeAritmatika *root, int tinggi, int level){
    if (root == NULL)
        return true;

    if (root->left == NULL && root->right == NULL)
        return (tinggi == level+1);

    if (root->left == NULL || root->right == NULL)
        return false;

    return isPerfectBinaryAritmatika(root->left, tinggi, level+1) &&
           isPerfectBinaryAritmatika(root->right, tinggi, level+1);
}
```

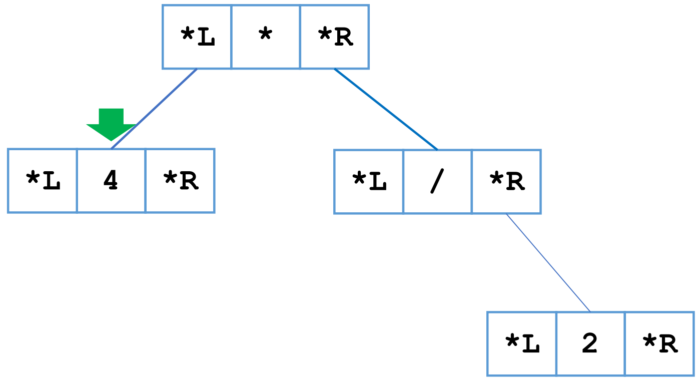
true &&

isPerfectBinaryAritmatika(root->left, 2, 1)

```
bool Tree::isPerfectBinaryAritmatika(NodeAritmatika *root, int tinggi, int level){
    if (root == NULL)
        return true;

    if (root->left == NULL && root->right == NULL)
        return (tinggi == level+1);

    if (root->left == NULL || root->right == NULL)
        return false;

    return isPerfectBinaryAritmatika(root->left, tinggi, level+1) &&
           isPerfectBinaryAritmatika(root->right, tinggi, level+1);
}
```

return (2==1); return true





isPerfectBinaryAritmatika(root, 2, 0) output false #2

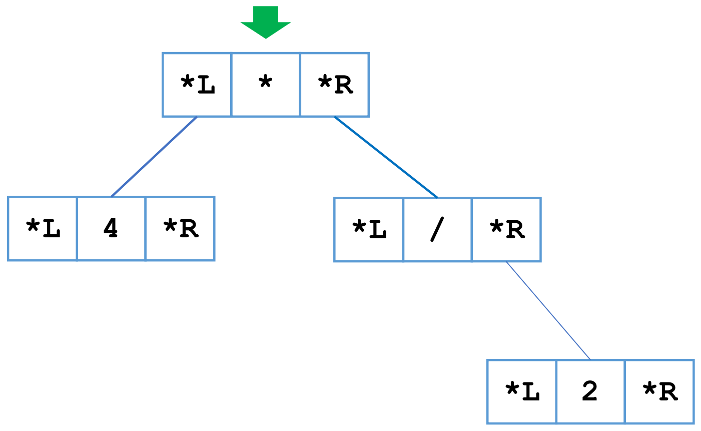
```
bool Tree::isPerfectBinaryAritmatika(NodeAritmatika *root, int tinggi, int level){
    if (root == NULL)
        return true;

    if (root->left == NULL && root->right == NULL)
        return (tinggi == level+1);

    if (root->left == NULL || root->right == NULL)
        return false;

    return isPerfectBinaryAritmatika(root->left, tinggi, level+1) &&
           isPerfectBinaryAritmatika(root->right, tinggi, level+1);
}
```

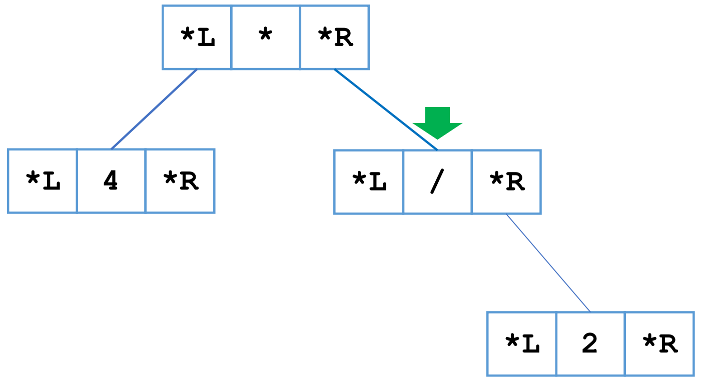
true && false

return false

isPerfectBinaryAritmatika(root->right, 2, 1)

```
bool Tree::isPerfectBinaryAritmatika(NodeAritmatika *root, int tinggi, int level){
    if (root == NULL)
        return true;

    if (root->left == NULL && root->right == NULL)
        return (tinggi == level+1);

    if (root->left == NULL || root->right == NULL)
        return false;                          return false

    return isPerfectBinaryAritmatika(root->left, tinggi, level+1) &&
           isPerfectBinaryAritmatika(root->right, tinggi, level+1);
}
```





Referensi

Utama:

1. Introduction to Algorithms 4th edition, Thomas H. Cormen, Charles E. Leiserson, Ronald E. Leiserson, Ronald L. Rivest, Clifford Stein, The MIT Press, 2022
2. Introduction to the Design and Analysis of Algorithms 3rd Edition, Anany Levitin, Pearson, 2011
3. Data Structures and Algorithms in C++, Michael T. Goodrich, Roberto Tamasia, David M. Mount, John Wiley & Sons, 2011

Pendukung:

1. Data Structures and Algorithms in C++ 4th Edition, Adam Drozdek, Cengage Learning, 2013

TERIMA KASIH

ANY QUESTIONS?