Tugas Praktikum Algoritma dan Struktur Data

Nama : Agil Deriansyah Hasan
Nim : 4522210125

Dosen Pengajar :

Dra.SRI REZEKI CANDRA NURSARI,M.Kom
Prak. Algoritma dan Struktur Data - I

S1-Teknik Informatika
Fakultas Teknik
Universitas Pancasila 2023/2024

cnthprak10-1.cpp | cnthprak10-2.cpp | pasd10-1.cpp

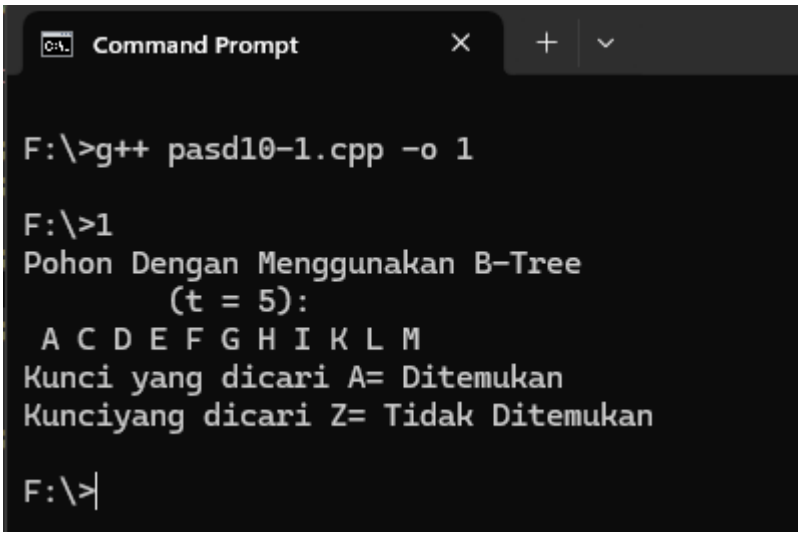
```cpp
#include <iostream>
using namespace std;

class btreenode {
    char *keys;
    int t, n;
    bool leaf;
    btreenode **children;

public:
    btreenode(int t, bool leaf);
    void insertNonFull(char k);
    void splitChild(int i, btreenode *y);
    void traverse();
    btreenode *search(char k);
    friend class btree;
};

class btree {
    int t;
    btreenode *root;

public:
    btree(int t) : t(t), root(nullptr) {}
    void traverse() { if (root != nullptr) root->traverse(); }
    btreenode* search(char k) { return (root == nullptr) ? nullptr : root->search(k); }
    void insert(char k);
};

btreenode::btreenode(int t1, bool leaf1) {
    t = t1;
    leaf = leaf1;
    keys = new char[2 * t - 1];
    children = new btreenode *[2 * t];
    n = 0;
}

void btreenode::traverse() {
    int i;
    for (i = 0; i < n; i++) {
        if (!leaf) {
            children[i]->traverse();
        }
        cout << " " << keys[i];
    }
    if (!leaf) {
        children[i]->traverse();
    }
}

```

cnthprak10-1.cpp | cnthprak10-2.cpp | pasd10-1.cpp

```cpp
btreenode *btreenode::search(char k) {
    int i = 0;
    while (i < n && k > keys[i])
        i++;
    if (keys[i] == k)
        return this;
    if (leaf)
        return nullptr;
    return children[i]->search(k);
}

void btree::insert(char k) {
    if (root == nullptr) {
        root = new btreenode(t, true);
        root->keys[0] = k;
        root->n = 1;
    } else {
        if (root->n == 2 * t - 1) {
            btreenode *s = new btreenode(t, false);
            s->children[0] = root;
            s->splitChild(0, root);
            int i = 0;
            if (s->keys[0] < k)
                i++;
            s->children[i]->insertNonFull(k);
            root = s;
        } else {
            root->insertNonFull(k);
        }
    }
}

void btreenode::insertNonFull(char k) {
    int i = n - 1;
    if (leaf) {
        while (i >= 0 && keys[i] > k) {
            keys[i + 1] = keys[i];
            i--;
        }
        keys[i + 1] = k;
        n++;
    } else {
        while (i >= 0 && keys[i] > k)
            i--;
        if (children[i + 1]->n == 2 * t - 1) {
            splitChild(i + 1, children[i + 1]);
            if (keys[i + 1] < k)
                i++;
        }
        children[i + 1]->insertNonFull(k);
    }
}
```

cnthprak10-1.cpp | cnthprak10-2.cpp | pasd10-1.cpp

```cpp
}

void btreenode::splitChild(int i, btreenode *y) {
    btreenode *z = new btreenode(y->t, y->leaf);
    z->n = t - 1;
    for (int j = 0; j < t - 1; j++)
        z->keys[j] = y->keys[j + t];
    if (!y->leaf) {
        for (int j = 0; j < t; j++)
            z->children[j] = y->children[j + t];
    }
    y->n = t - 1;
    for (int j = n; j >= i + 1; j--)
        children[j + 1] = children[j];
    children[i + 1] = z;
    for (int j = n - 1; j >= i; j--)
        keys[j + 1] = keys[j];
    keys[i] = y->keys[t - 1];
    n++;
}

int main() {
    btree t(3);
    t.insert('F');
    t.insert('L');
    t.insert('A');
    t.insert('G');
    t.insert('K');
    t.insert('M');
    t.insert('C');
    t.insert('D');
    t.insert('E');
    t.insert('H');
    t.insert('I');

    cout << "Pohon Dengan Menggunakan B-Tree" << endl;
    cout << "        (t = 5):" << endl;
    t.traverse();
    cout << endl;

    char k = 'A';
    (t.search(k) != nullptr) ? cout << "Key " << k << " is found." : cout << "Key " << k << " is not found.";
    cout << endl;
    k = 'Z';
    (t.search(k) != nullptr) ? cout << "Key " << k << " is found." : cout << "Key " << k << " is not found.";
    cout << endl;

    return 0;
}
```



Pseudocode :

```
Kamus/Deklarasi Variabel fungsi btreenode
t1 = int
leaf1 = bool

Algoritma/Deskripsi fungsi btreenode(int t1, bool leaf1)
t = t1
leaf = leaf1
keys = new int[2*t-1]
children = new btreenode *[2*t]
n=0

Kamus/Deklarasi Variabel fungsi traverse
i = int

Algoritma/Deskripsi fungsi traverse
for (i=0;i<n;i++)
        if(!leaf)
                children[i]->traverse()
                keys[i]
endfor
        if(!leaf)
        children[i]->traverse()
        endif

Kamus/Deklarasi variabel fungsi search
i = int
c = char

Algoritma/Deskripsi fungsi search (k)
while(i<n && k>keys[i])
        i++
if(keys[i]==k)
        return this
if(leaf == true)
        return NULL
return c[i]->search(k)
```

```
Kamus/Deklarasi Variabel fungsi insert
k = char
i = int

Algoritma/Deskripsi fungsi insert(k)
if(root == NULL)
        root = new btreenode(t,true)
        root->keysi[0]=k
        root->n=1
else
        if(root->n==2*t-1)
                btreenode *s =new btreenode(t,false)
                s->children[0]=root
                s->splitchildren(0,root)
                i = 0
                        if(s->keys[0]<k)
                                i++
                                s->children[i]->sisiptdkpenuh(k)
                                root=s
                else
                        root->sisiptdkpenuh(k)
endif
endif
```

```
Kamus/Deklarasi Variabel fungsi insertnonfull
i = int
k = char

Algoritma/Deskripsi fungsi sisiptdkpenuh(k)
i=n-1
if(leaf)
        while(i>=0 && keys[i]>k)
                keys[i+1]=keys[i]
                i--
        endwhile
        keys[i+1]=k
        n++
else
    while(i>=0&&keys[i]>k)
            i--
    if(children[i+1]->n==2*t-1)
                splitchildren(i+1, children[i+1])
                if(keys[i+1]<k)
                        i++
    endif
            c[i+1]->insertnonfull(k)
```

```
Kamus/Deklarasi Variabel fungsi splitchild
i,j=int

Algoritma/Deskripsi fungsi splitanak(i,btreenode *y)
btreenode *z = new btreenode(y->t,y->leaf);
        z->n=t-1;
        for(int j=0;j<t-1;j++)
                z->keys[j]=y->keys[j+t]
        if(y->leaf==false)
                for(int j=0;j<t;j++)
                        z->children[j]=y->children[j+t]
        endif
        y->n=t-1
        for(int j=n;j>=i+1;j--)
                children[j+1]=children[j]
                children[i+1]=z
        for(int j=n-1;j>=i;j--)
                keys[j+1]=keys[j]
                keys[i]=y->keys[t-1]
                n=n+1
```

```
Kamus/Deklarasi Variabel fungsi utama
k = char

Algoritma/Deskripsi fungsi utama
    btree t(3);
    t.insert('F');
    t.insert('L');
    t.insert('A');
    t.insert('G');
    t.insert('K');
    t.insert('M');
    t.insert('C');
    t.insert('D');
    t.insert('E');
    t.insert('H');
    t.insert('I');
k='A'
(t.search(k) != NULL)?; print k ; print k
k = 'Z'
(t.search(k) != NULL)?; print k ; print k
return 0
```

Algoritma :

1. Membuat fungsi btreenode(t1,leaf1)
2. t = t1
3. leaf = leaf1
4. keys = new int[2*t-1]
5. c = new btreenode *[2*t]
6. n=0
7. membuat fungsi traverse
8. Selama (i=0) maka kerjakan baris 9 s.d 12
9. Jika (leaf==false)
10. children[i] -> traverse
11. Mencetak Nilai kunci[i]
12. i++
13. Jika (leaf==false)
14. children[i]->traverse()
15. Membuat fungsi search(k)
16. i=0
17. Selama (i<n && k>keys[i])
18. i++
19. Jika (keys[i]==k)
20. return this
21. Jika (leaf == true)
22. return NULL
23. return children[i]->search(k)
24. Membuat fungsi sisip (k)
25. Jika (root == NULL) maka kerjakan baris 26 s.d 28 kalau tidak 29 s.d 38
26. root = new btreenode(t,true)
27. root->keys[0]=k
28. root->n=1
29. Jika (root->n==2*t-1) maka kerjakan baris 30 s.d 37 kalau tidak baris 38
30. btreenode *s =new btreenode(t,false)
31. s->children[0]=root
32. s->splitchildren(0,root)
33. i = 0
34. Jika (s->keys[0]<k)
35. i++
36. s->c[i]->insertnofull(k)
37.root=s
38. root->insertnofull(k)
39. Membuat fungsi insertnofull (k)
40. i=n-1
41. Jika(leaf==true) maka kerjakan baris 42 s.d 46 kalau tidak 47 s.d
42. Selama (i>=0 && keys[i]>k) 43 s.d 44
43. keyss[i+1]=keys[i];
44. i--
45. keys[i+1]=k
46. n=n+1
47. Selama (i>=0&&keys[i]>k)
48. i–
49. Jika (children[i+1]->n==2*t-1) maka kerjakan baris 50 s.d 52
50. splitchildren(i+1, children[i+1])
51. Jika(keys[i+1]<k)
52. i++
53. children[i+1]->insertnofull(k)

54. Membuat fungsi splitanak(i,*y)
55. btreenode *z = new btreenode(y->t,y->leaf)
56. z->n=t-1
57. Selama (j=0)
58. z->kunci[j]=y->kunci[j+t]
59. j++
60. Jika (y->leaf==false) maka kerjakan baris 61 s.d 62
61. Selama (j=0)
62. z->c[j]=y->c[j+t]
63. y->n=t-1
64. Selama (j=n)
65. c[j+1]=c[j]
66. c[i+1]=z
67. j–
68. Selama (j=n-1)
69. kunci[j+1]=kunci[j]
70. kunci[i]=y->kunci[t-1]
71. n=n+1
72. j–
73. Membuat fungi utama
74. Membuat class btreenode
75. Mendeklarasikan class btreenode dengan kata kunci public
76. Membuat class btree
77. Mendeklarasikan class btree dengan kata kunci public
btree t(3);
78. t.insert('F');
79. t.insert('L');
80. t.insert('A');
81. t.insert('G');
82. t.insert('K');
83. t.insert('M');
84. t.insert('C');
85. t.insert('D');
86. t.insert('E');
87. t.insert('H');
88. t.insert('I');

89. k=A
90. (t.search(k) != NULL)?
91. Mencetak nilai k
92. k=Z
93. (t.search(k) != NULL)?
94. Mencetak nilai k
95. Selesai