Contoh Praktikum Algoritma dan Struktur Data

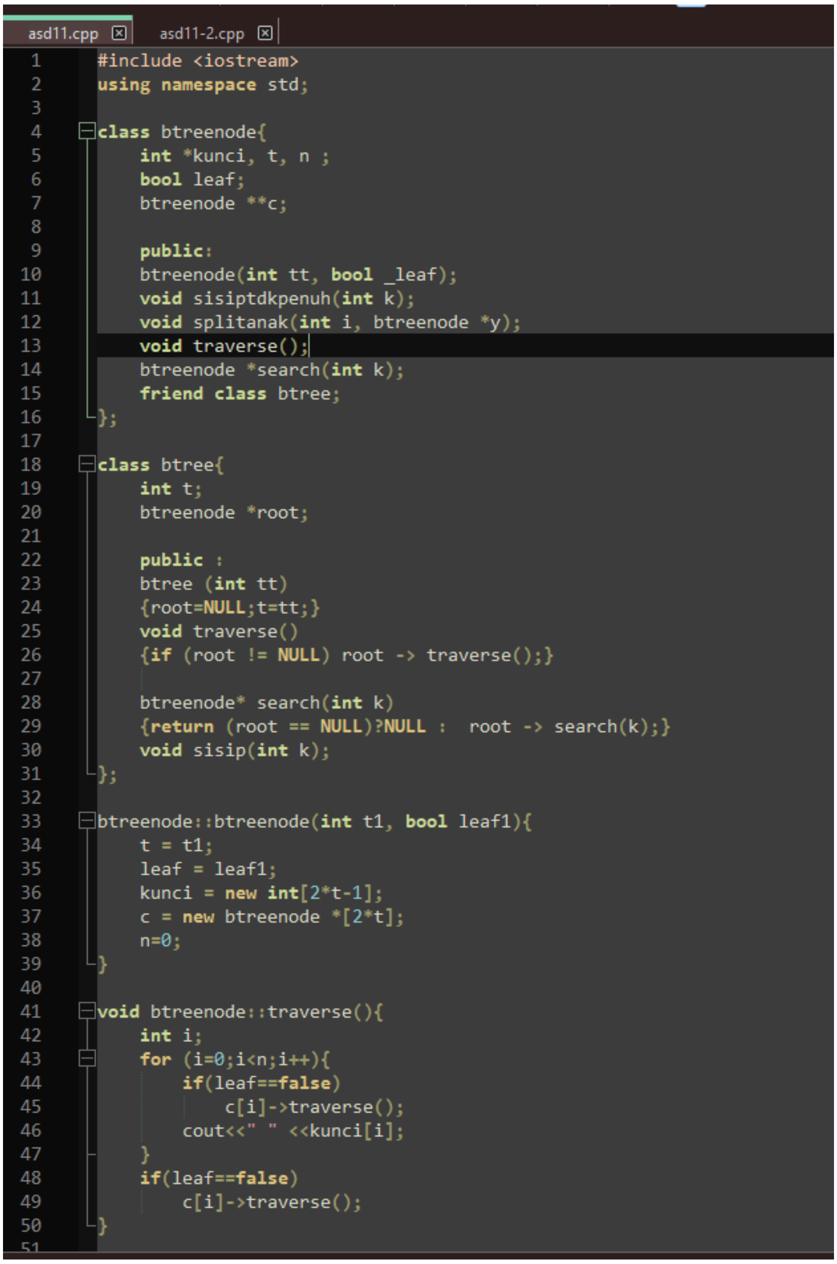
Nama : Agil Deriansyah Hasan
Nim : 4522210125

Dosen Pengajar :

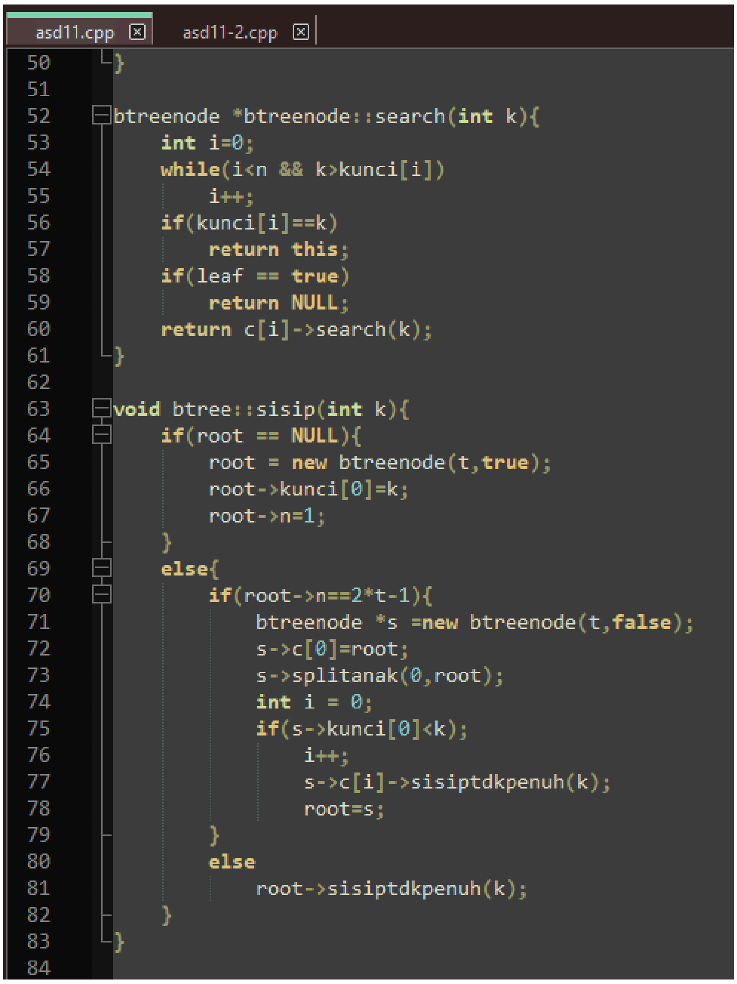
Dra.SRI REZEKI CANDRA NURSARI,M.Kom
Prak. Algoritma dan Struktur Data - I

S1-Teknik Informatika
Fakultas Teknik
Universitas Pancasila 2023/2024

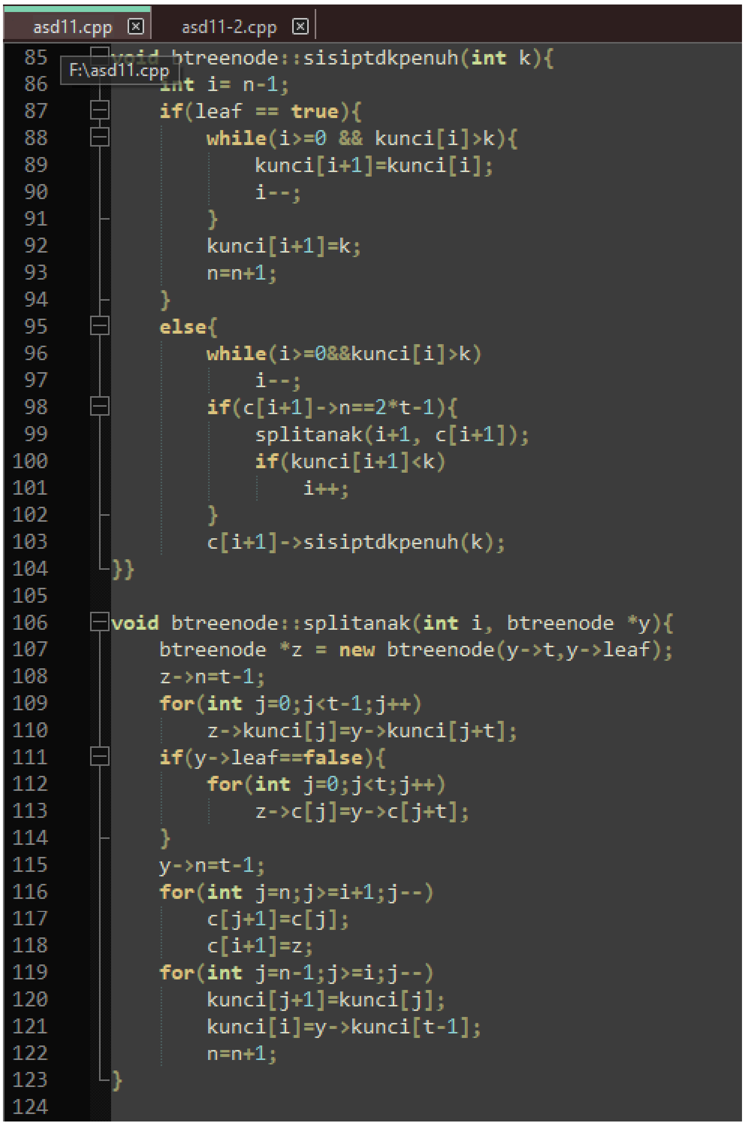
cnthprak10-1



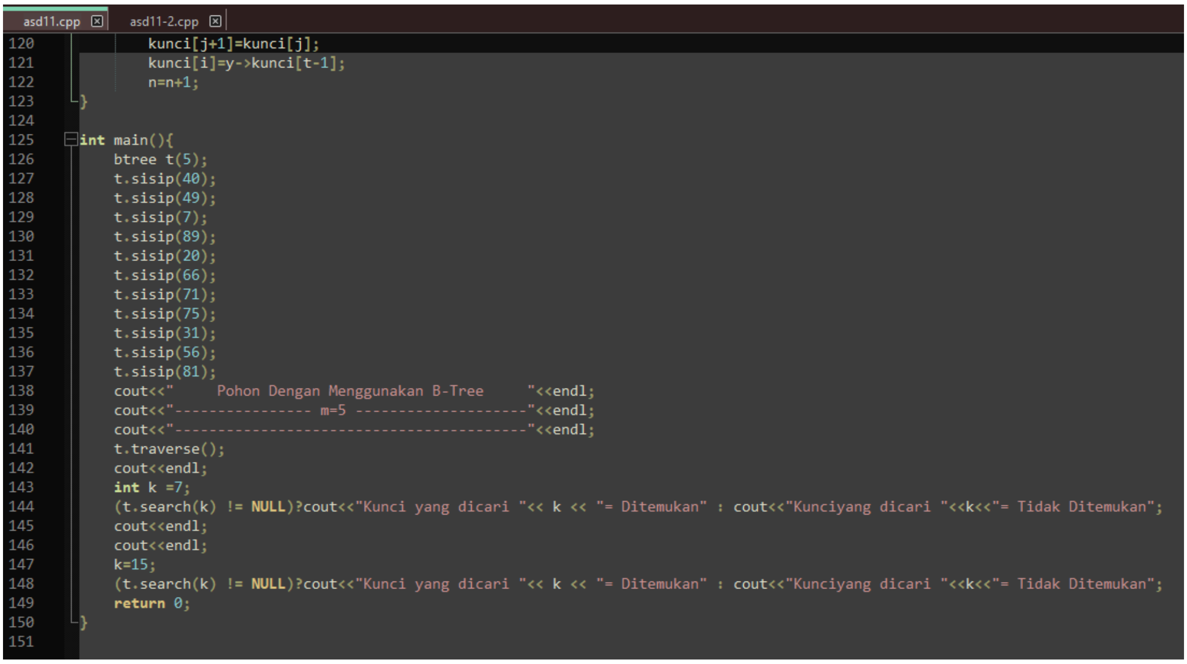
```
#include <iostream>
using namespace std;

class btreenode{
    int *kunci, t, n ;
    bool leaf;
    btreenode **c;

    public:
    btreenode(int tt, bool _leaf);
    void sisiptdkpenuh(int k);
    void splitanak(int i, btreenode *y);
    void traverse();
    btreenode *search(int k);
    friend class btree;
};

class btree{
    int t;
    btreenode *root;

    public :
    btree (int tt)
    {root=NULL;t=tt;}
    void traverse()
    {if (root != NULL) root -> traverse();}

    btreenode* search(int k)
    {return (root == NULL)?NULL :  root -> search(k);}
    void sisip(int k);
};

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

void btreenode::traverse(){
    int i;
    for (i=0;i<n;i++){
        if(leaf==false)
            c[i]->traverse();
        cout<<" " <<kunci[i];
    }
    if(leaf==false)
        c[i]->traverse();
}
```



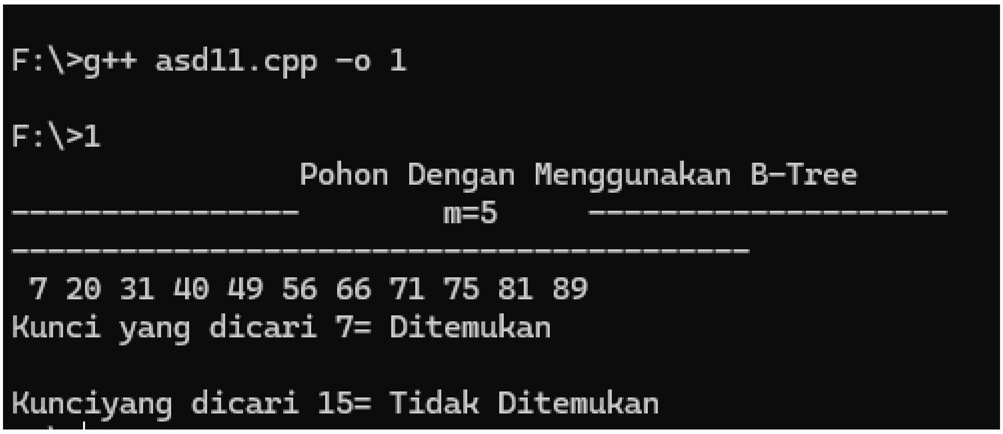
```
}

btreenode *btreenode::search(int k){
    int i=0;
    while(i<n && k>kunci[i])
        i++;
    if(kunci[i]==k)
        return this;
    if(leaf == true)
        return NULL;
    return c[i]->search(k);
}

void btree::sisip(int k){
    if(root == NULL){
        root = new btreenode(t,true);
        root->kunci[0]=k;
        root->n=1;
    }
    else{
        if(root->n==2*t-1){
            btreenode *s =new btreenode(t,false);
            s->c[0]=root;
            s->splitanak(0,root);
            int i = 0;
            if(s->kunci[0]<k);
                i++;
                s->c[i]->sisiptdkpenuh(k);
                root=s;
        }
        else
            root->sisiptdkpenuh(k);
    }
}
```



```cpp
void btreenode::sisiptdkpenuh(int k){
    int i= n-1;
    if(leaf == true){
        while(i>=0 && kunci[i]>k){
            kunci[i+1]=kunci[i];
            i--;
        }
        kunci[i+1]=k;
        n=n+1;
    }
    else{
        while(i>=0&&kunci[i]>k)
            i--;
        if(c[i+1]->n==2*t-1){
            splitanak(i+1, c[i+1]);
            if(kunci[i+1]<k)
                i++;
        }
        c[i+1]->sisiptdkpenuh(k);
}}

void btreenode::splitanak(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->kunci[j]=y->kunci[j+t];
    if(y->leaf==false){
        for(int j=0;j<t;j++)
            z->c[j]=y->c[j+t];
    }
    y->n=t-1;
    for(int j=n;j>=i+1;j--)
        c[j+1]=c[j];
        c[i+1]=z;
    for(int j=n-1;j>=i;j--)
        kunci[j+1]=kunci[j];
        kunci[i]=y->kunci[t-1];
        n=n+1;
}

```



```cpp
        kunci[j+1]=kunci[j];
        kunci[i]=y->kunci[t-1];
        n=n+1;
}

int main(){
    btree t(5);
    t.sisip(40);
    t.sisip(49);
    t.sisip(7);
    t.sisip(89);
    t.sisip(20);
    t.sisip(66);
    t.sisip(71);
    t.sisip(75);
    t.sisip(31);
    t.sisip(56);
    t.sisip(81);
    cout<<"     Pohon Dengan Menggunakan B-Tree     "<<endl;
    cout<<"---------------- m=5 --------------------"<<endl;
    cout<<"-----------------------------------------"<<endl;
    t.traverse();
    cout<<endl;
    int k =7;
    (t.search(k) != NULL)?cout<<"Kunci yang dicari "<< k << "= Ditemukan" : cout<<"Kunciyang dicari "<<k<<"= Tidak Ditemukan";
    cout<<endl;
    cout<<endl;
    k=15;
    (t.search(k) != NULL)?cout<<"Kunci yang dicari "<< k << "= Ditemukan" : cout<<"Kunciyang dicari "<<k<<"= Tidak Ditemukan";
    return 0;
}

```



```
F:\>g++ asd11.cpp -o 1

F:\>1
                Pohon Dengan Menggunakan B-Tree
----------------        m=5     --------------------
-----------------------------------------
 7 20 31 40 49 56 66 71 75 81 89
Kunci yang dicari 7= Ditemukan

Kunciyang dicari 15= Tidak Ditemukan
```

Pseudocode :

```
Kamus/Deklarasi variabel class btreenode
*kunci, t, n, tt, i,k = int
leaf = bool
btree = friend class

Kamus/Deklarasi variabel class btree
t,tt,k = int

Algoritma/Deskripsi class btree
public :
{root=NULL;t=tt;}
{if (root != NULL) root -> traverse();}
btreenode* search(int k)
{return (root == NULL)?NULL :  root -> search(k);}

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

Algoritma/Deskripsi fungsi btreenode(int t1, bool leaf1)
t = t1
leaf = leaf1
kunci = new int[2*t-1]
c = 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==false)
        c[i]->traverse()
        kunci[i]
endfor
    if(leaf==false)
    c[i]->traverse()
```

```
Kamus/Deklarasi variabel fungsi search
k, i = int

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

Kamus/Deklarasi Variabel fungsi sisip
k = int

Algoritma/Deskripsi fungsi sisip
if(root == NULL)
    root = new btreenode(t,true)
    root->kunci[0]=k
    root->n=1
else
    if(root->n==2*t-1)
        btreenode *s =new btreenode(t,false)
        s->c[0]=root
        s->splitanak(0,root)
        i = 0
            if(s->kunci[0]<k)
                i++
                s->c[i]->sisiptdkpenuh(k)
                root=s
        else
            root->sisiptdkpenuh(k)
endif
endif
```

```
Kamus/Deklarasi Variabel fungsi sisiptdkpenuh
i , k = int

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

```
Kamus/Deklarasi Variabel fungsi splitanak
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->kunci[j]=y->kunci[j+t]
    if(y->leaf==false)
        for(int j=0;j<t;j++)
            z->c[j]=y->c[j+t]
    endif
    y->n=t-1
    for(int j=n;j>=i+1;j--)
        c[j+1]=c[j]
        c[i+1]=z
    for(int j=n-1;j>=i;j--)
        kunci[j+1]=kunci[j]
        kunci[i]=y->kunci[t-1]
        n=n+1

Kamus/Deklarasi Variabel fungsi utama
k = int

Algoritma/Deskripsi fungsi utama
btree t(5);
    t.sisip(40);
    t.sisip(49);
    t.sisip(7);
    t.sisip(89);
    t.sisip(20);
    t.sisip(66);
    t.sisip(71);
    t.sisip(75);
    t.sisip(31);
    t.sisip(56);
    t.sisip(81);
    t.traverse()
```

```
k=7
(t.search(k) != NULL)?; print k ; print k
k = 15
(t.search(k) != NULL)?; print k ; print k
return 0
```

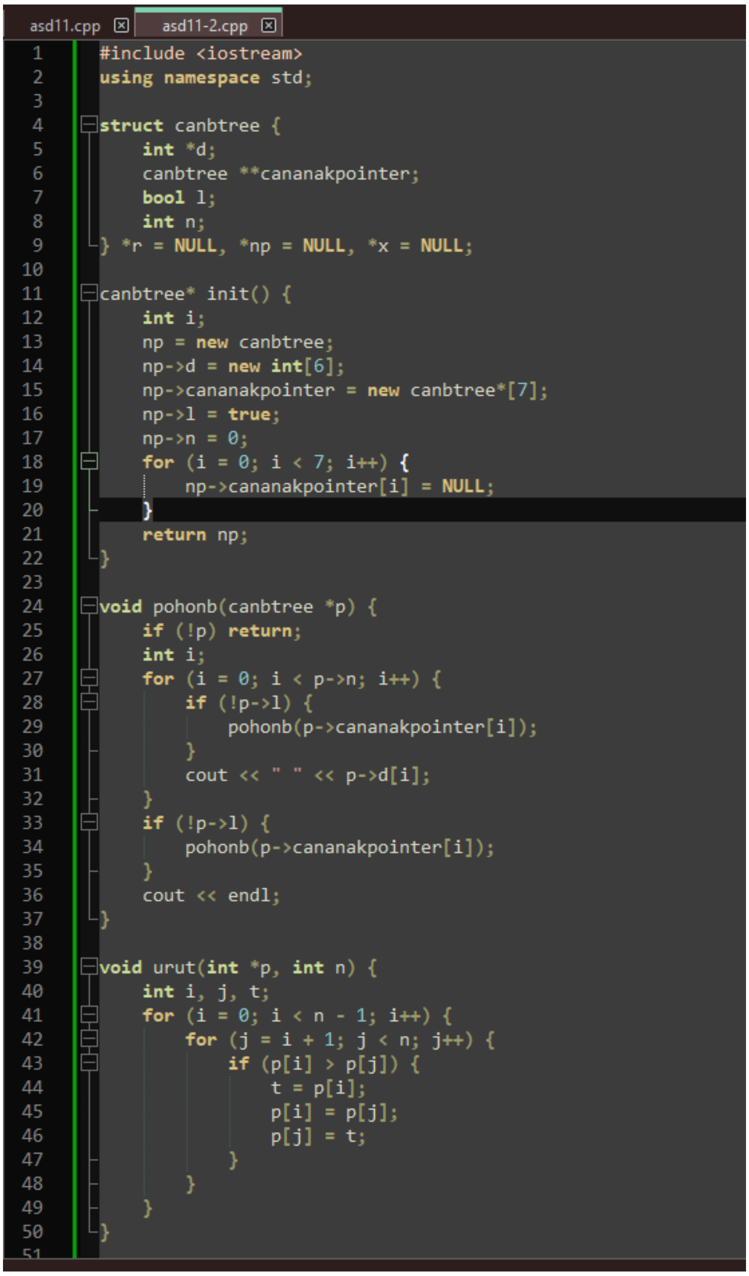
Algoritma :

```
1.  Membuat fungsi btreenode(t1,leaf1)
2.  t = t1
3.  leaf = leaf1
4.  kunci = 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. c[i] -> traverse
11. Mencetak Nilai kunci[i]
12. i++
13. Jika (leaf==false)
14. c[i]->traverse()
15. Membuat fungsi search(k)
16. i=0
17. Selama (i<n && k>kunci[i])
18. i++
19. Jika (kunci[i]==k)
20. return this
21. Jika (leaf == true)
22. return NULL
23. return c[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->kunci[0]=k
28. root->n=1
```

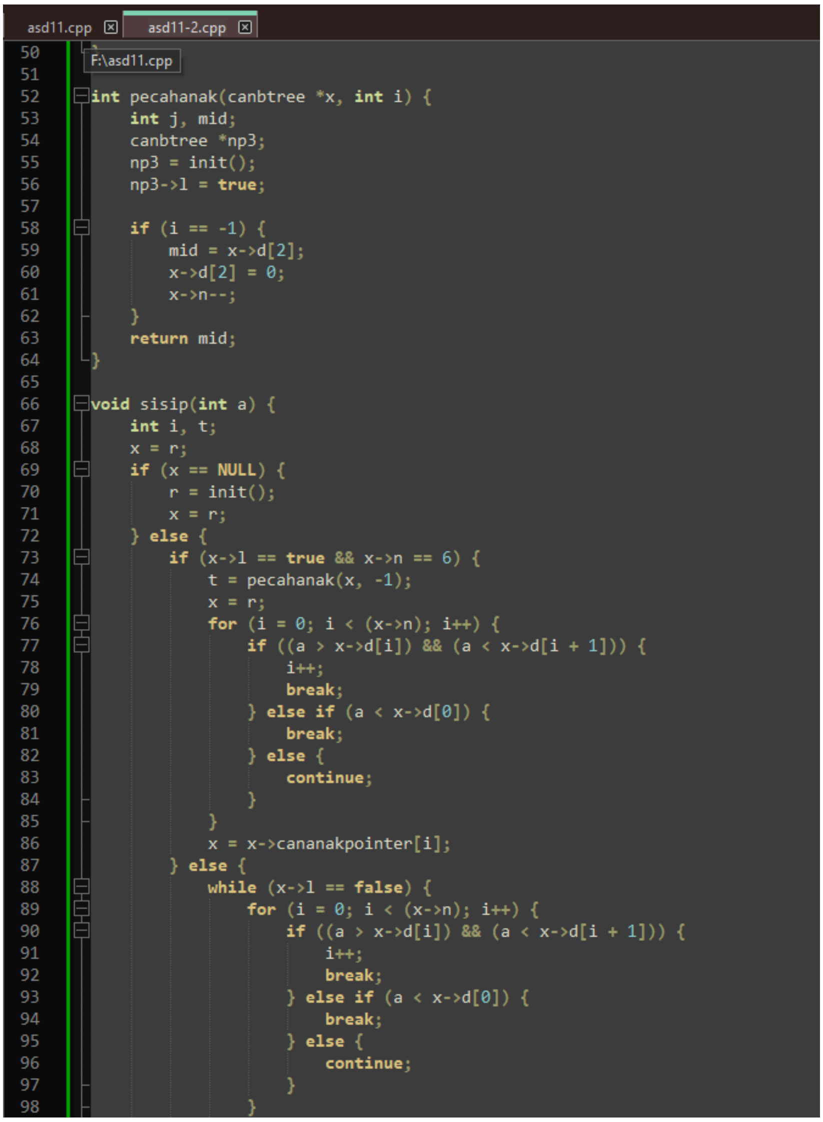
```
29. Jika (root->n==2*t-1) maka kerjakan baris
30 s.d 37 kalau tidak baris 38
30. btreenode *s =new btreenode(t,false)
31. s->c[0]=root
32. s->splitanak(0,root)
33. i = 0
34. Jika (s->kunci[0]<k)
35. i++
36. s->c[i]->sisiptdkpenuh(k)
37.root=s
38. root->sisiptdkpenuh(k)
39. Membuat fungsi sisiptdkpenuh (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 && kunci[i]>k) 43 s.d 44
43. kunci[i+1]=kunci[i];
44. i--
45. kunci[i+1]=k
46. n=n+1
47. Selama (i>=0&&kunci[i]>k)
48. i–
49. Jika (c[i+1]->n==2*t-1) maka kerjakan
baris 50 s.d 52
50. splitanak(i+1, c[i+1])
51. Jika(kunci[i+1]<k)
52. i++
53. c[i+1]->sisiptdkpenuh(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 btree t(5)
75. t.sisip(40)
76. t.sisip(49)
77. t.sisip(7)
78. t.sisip(89)
79. t.sisip(20)
80. t.sisip(66)
81. t.sisip(71)
82. t.sisip(75)
83. t.sisip(31)
84. t.sisip(56)
85. t.sisip(81)
86. Memanggil fungsi t.traverse()
87. k=7
88. (t.search(k) != NULL)?
89. Mencetak nilai k
90. k=15
91. (t.search(k) != NULL)?
92. Mencetak nilai k
93. Selesai
```

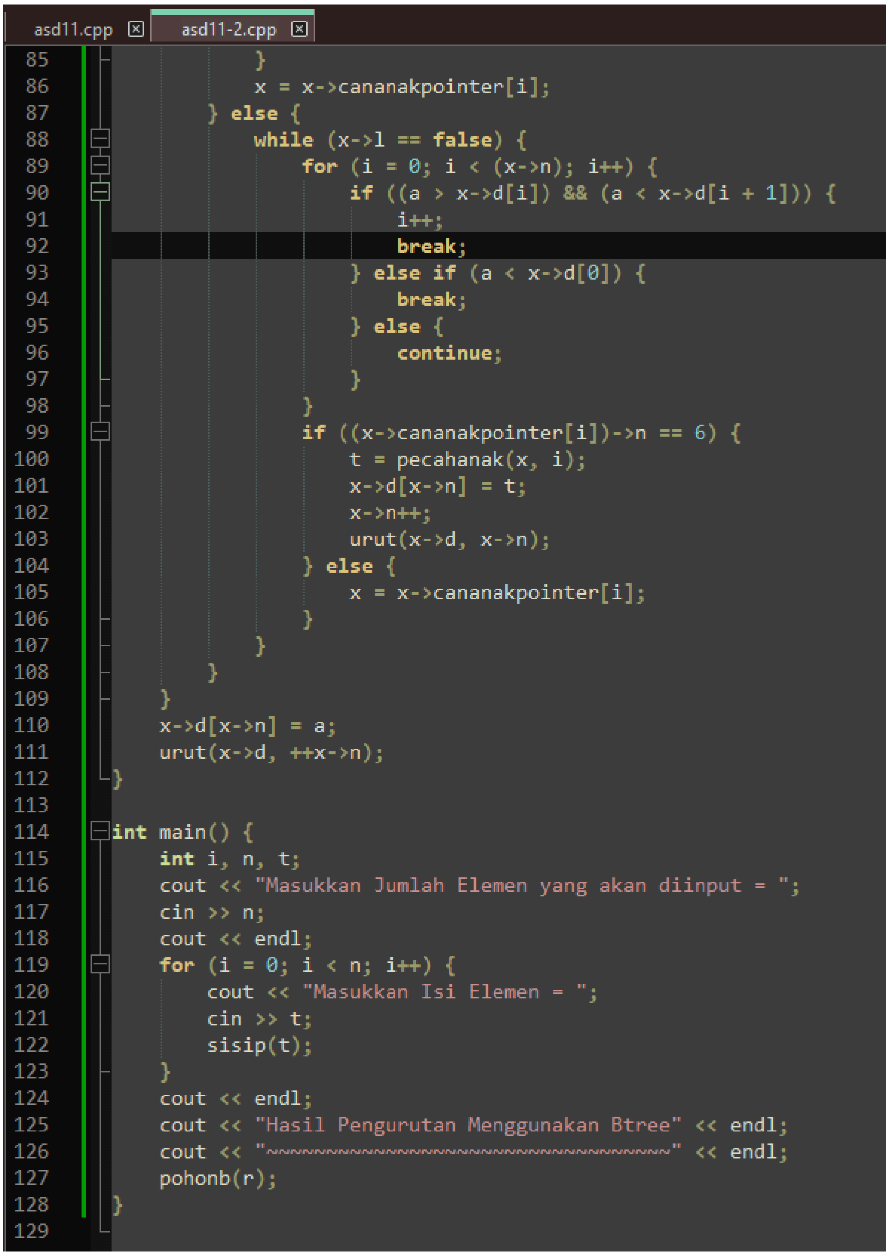
cnthprak10-2



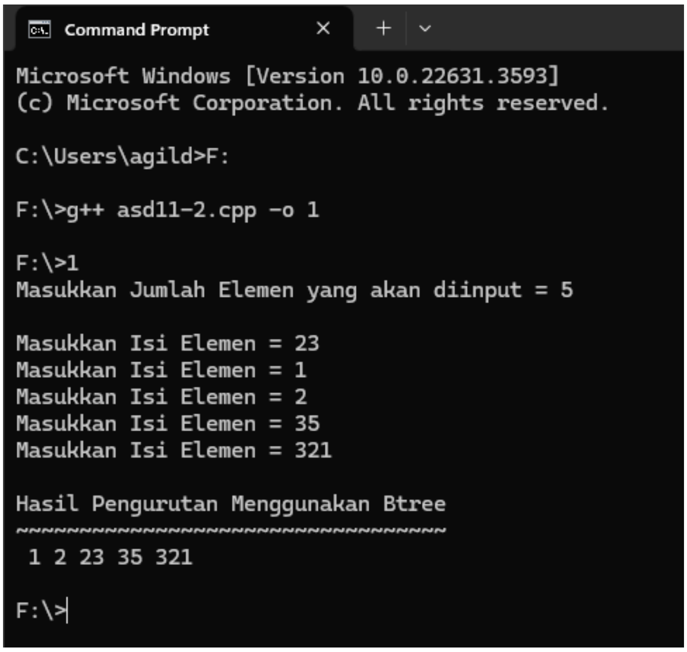
```cpp
#include <iostream>
using namespace std;

struct canbtree {
    int *d;
    canbtree **canakpointer;
    bool l;
    int n;
} *r = NULL, *np = NULL, *x = NULL;

canbtree* init() {
    int i;
    np = new canbtree;
    np->d = new int[6];
    np->canakpointer = new canbtree*[7];
    np->l = true;
    np->n = 0;
    for (i = 0; i < 7; i++) {
        np->canakpointer[i] = NULL;
    }
    return np;
}

void pohonb(canbtree *p) {
    if (!p) return;
    int i;
    for (i = 0; i < p->n; i++) {
        if (!p->l) {
            pohonb(p->canakpointer[i]);
        }
        cout << " " << p->d[i];
    }
    if (!p->l) {
        pohonb(p->canakpointer[i]);
    }
    cout << endl;
}

void urut(int *p, int n) {
    int i, j, t;
    for (i = 0; i < n - 1; i++) {
        for (j = i + 1; j < n; j++) {
            if (p[i] > p[j]) {
                t = p[i];
                p[i] = p[j];
                p[j] = t;
            }
        }
    }
}
```



```cpp
}

int pecahanak(canbtree *x, int i) {
    int j, mid;
    canbtree *np3;
    np3 = init();
    np3->l = true;

    if (i == -1) {
        mid = x->d[2];
        x->d[2] = 0;
        x->n--;
    }
    return mid;
}

void sisip(int a) {
    int i, t;
    x = r;
    if (x == NULL) {
        r = init();
        x = r;
    } else {
        if (x->l == true && x->n == 6) {
            t = pecahanak(x, -1);
            x = r;
            for (i = 0; i < (x->n); i++) {
                if ((a > x->d[i]) && (a < x->d[i + 1])) {
                    i++;
                    break;
                } else if (a < x->d[0]) {
                    break;
                } else {
                    continue;
                }
            }
            x = x->canakpointer[i];
        } else {
            while (x->l == false) {
                for (i = 0; i < (x->n); i++) {
                    if ((a > x->d[i]) && (a < x->d[i + 1])) {
                        i++;
                        break;
                    } else if (a < x->d[0]) {
                        break;
                    } else {
                        continue;
                    }
                }
```



```cpp
            }
            x = x->cananakpointer[i];
        } else {
            while (x->l == false) {
                for (i = 0; i < (x->n); i++) {
                    if ((a > x->d[i]) && (a < x->d[i + 1])) {
                        i++;
                        break;
                    } else if (a < x->d[0]) {
                        break;
                    } else {
                        continue;
                    }
                }
                if ((x->cananakpointer[i])->n == 6) {
                    t = pecahanak(x, i);
                    x->d[x->n] = t;
                    x->n++;
                    urut(x->d, x->n);
                } else {
                    x = x->cananakpointer[i];
                }
            }
        }
    }
    x->d[x->n] = a;
    urut(x->d, ++x->n);
}

int main() {
    int i, n, t;
    cout << "Masukkan Jumlah Elemen yang akan diinput = ";
    cin >> n;
    cout << endl;
    for (i = 0; i < n; i++) {
        cout << "Masukkan Isi Elemen = ";
        cin >> t;
        sisip(t);
    }
    cout << endl;
    cout << "Hasil Pengurutan Menggunakan Btree" << endl;
    cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
    pohonb(r);
}
```



```
Microsoft Windows [Version 10.0.22631.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Users\agild>F:

F:\>g++ asd11-2.cpp -o 1

F:\>1
Masukkan Jumlah Elemen yang akan diinput = 5

Masukkan Isi Elemen = 23
Masukkan Isi Elemen = 1
Masukkan Isi Elemen = 2
Masukkan Isi Elemen = 35
Masukkan Isi Elemen = 321

Hasil Pengurutan Menggunakan Btree
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 1 2 23 35 321

F:\>
```

Algoritma :

```
1.  Membuat fungsi canbtree* init
2.  Deklarasi struktur ( struct { *d,l,n,**cannakpointer)
3.  Mendefinisikan *r = NULL, *np = NULL, *x = NULL
4.  np = new canbtree
5.  np->d = new int[6]
6.  np->cananakpointer = new canbtree*[7]
7.  np->l = true
8.  np->n = 0
9.  Selama ( i = 0) maka kerjakan baris 10 s.d 11
10. np ->cananakpointer[i]=NULL
11. i++
12. returnnp
13. Membuat fungsi pohonb(canbtree *p)
14. Selama ( i=0) maka kerjakan baris 15 s.d 20
15. Jika (p->==false) maka kerjakan baris 16
16. pohonB(p->cananakpointer[i])
17. Mencetak p -> d[i]
18. Jika (p->l==false) maka kerjakan baris 19
19. pohonB(p->cananakpointer[i])
20. i++
21. Membuat fungsi urut(*p, n)
22. Selama (i=0) maka kerjakan baris 23 s.d 29
23. selama (j=i) maka kerjakan baris 24 s.d 28
24. Jika (p[i] > p[j]) maka kerjakan baris 25 s.d 27
25. t = p[i]
26. p[i] = p[j]
27. p[j] = t
28. j++
29. i++
30. Membuat fungsi pecahanak(canbtree *x, i)
31. canbtree *np3
32. np3=init
33. np3->l=true
34. Jika (i==-1) maka kerjakan baris 35 s.d 37
35. mid = x->d[2]
36. x->d[2] = 0
37. x->n--
38. return mid
39. Membuat fungsi sisip (a)
40. x=r
41. Jika (x==NULL) maka kerjakan baris 42 s.d 43 kalau tidak 44 s.d 70
42. r=init
43. x=r
44. Jika(x->l == true && x->n == 6) maka kerjakan baris 45 s.d 55 kalau tidak baris 56 s.d 70
45. t = pecahanak(x, -1)
46. x = r
47. selama (i=0) maka kerjakan baris 48 s.d 54
48. Jika ((a > x->d[i]) && (a < x->d[i + 1])) maka kerjakan baris 49 s.d 50 kalau tdak 51 s.d 53
49. i++
50. break
51. Jika (a < x->d[0]) maka kerjakan baris 52 kalau tidak baris 53
52. break
53. continue
54. i++
55. x = x->cananakpointer[i]
56. Selama (x->1==false) maka kerjakan baris 57 s.d 6
57. Selama (i=0) maka kerjakan baris 58 s.d 64
58. Jika ((a > x->d[i]) && (a < x->d[i + 1])) maka kerjakan baris 59 s.d 60 kalu tidak 61 s.d  63
59. i++
60. break
61. jika(a < x->d[0]) maka kerjakan baris 62 kalau tidak 63
62. break
63. continue
64. i++
65. Jika ((x->cananakpointer[i])->n == 6) maka kerjakan baris 66 s.d 69 kalau tidak 70
66. t = pecahanak(x, i)
67. x->d[x->n] = t
68. x->n++
69. urut(x->d, x->n)
70. x = x->cananakpointer[i]
```

```
71.  x->d[d->n]=a
72. urut(x->d,x->n)
73. x->n++
74. Membuat fungsi utama
75. Menginput/Memasukkan nilai n
76. Selama (i=0) maka kerjakan baris 77 s.d 79
77. Menginput/Memasukkan nilai t
78. Memanggil fungsi sisip(t)
79. i++
80. Memanggil fungsi pohonb(r)
```

Pseudocode

```
Kamus/Deklarasi Variabel fungsi init
i = int

Algoritma/Deklarasi fungsi init
np = new canbtree
np->d = new int[6]
np->cananakpointer = new canbtree*[7]
np->l = true
np->n = 0
for (i = 0; i < 7; i++)
        np->cananakpointer[i] = NULL
endfor
return np

Kamus/Deklarasi Variabel fungsi pohonb
*p = canbtree
i = int

Algortima/Deklarasi fungi pohonb(canbtree *p)
if (!p) return
        for (i = 0; i < p->n; i++)
                if (!p->l)
                pohonb(p->cananakpointer[i])
        endfor
        print p->d[i]
endif
if (!p->l)
pohonb(p->cananakpointer[i])
endif
```

```
Kamus/Deklarasi variabel fungsi urut
*p, n, i, j, t =int

Algortima/Deskripsi fungsi urut (*p,  n)
for (i = 0; i < n - 1; i++)
        for (j = i + 1; j < n; j++)
                if (p[i] > p[j])
                t = p[i]
                p[i] = p[j]
                p[j] = t
                endif
        endfor
endfor

Kamus/Deklarasi Variabel fungsi pecahanak
*x = canbtree
i,j,mid = int

Algoritma/Deskripsi fungsi pecah anak (canbtree *x, i)
canbtree *np3
np3 = init()
np3->l = true
        if (i == -1)
        mid = x->d[2]
        x->d[2] = 0
        x->n–
        endif
return mid
```

```
Kamus/Deklarasi Variabel fungsi sisip
a, i, t = int
```

```
Algoritma/Deskripsi fungsi sisip (a)
x = r
if (x == NULL)
    r = init()
    x = r
else
if (x->l == true && x->n == 6)
    t = pecahanak(x, -1)
    x = r
    for (i = 0; i < (x->n); i++)
        if ((a > x->d[i]) && (a < x->d[i + 1]))
        i++
        break
        else if (a < x->d[0])
        break
        else
        continue
        endif
        endif
    endfor
        x = x->cananakpointer[i]
```

```
else
    while (x->l == false)
    for (i = 0; i < (x->n); i++)
        if ((a > x->d[i]) && (a < x->d[i + 1]))
        i++
        break
        else if (a < x->d[0])
        break
        else
        continue
    endfor
            if ((x->cananakpointer[i])->n == 6)
            t = pecahanak(x, i)
            x->d[x->n] = t
            x->n++
            urut(x->d, x->n)
            else
            x = x->cananakpointer[i]
            x->d[x->n] = a
            urut(x->d, ++x->n)
            endif
    endwhile
endif
endif
```

```
Kamus/Deklarasi Variabel fungsi utama
i,n,t = int

Algoritma/Deskripsi fungsi utama
input n
for (i = 0; i < n; i++)
    input t
    sisip(t)
endfor
pohonb(r)
```