

TUGAS JOBSHEET 07
PRAKTIKUM STRUKTUR DATA



DOSEN PENGAMPU:
Vera Irma Delianti, S.Pd., M.Pd.T.

OLEH:
M. Ilham
23343008

PROGRAM STUDI INFORMATIKA
DEPARTEMEN TEKNIK ELEKTRONIKA
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2024

1. B-tree.c

a. SOURCE CODE

```
/* Nama file   : B-tree.c
Pembuat       : M. Ilham 23343008
Tgl pembuatan : 28 Maret 2024*/

#include <stdio.h>
#include <stdlib.h>

#define MAX 3
#define MIN 2

struct BTreeNode {
    int val[MAX + 1], count;
    struct BTreeNode *link[MAX + 1];
};

struct BTreeNode *root;

struct BTreeNode *createNode(int val, struct
BTreeNode *child) {
    struct BTreeNode *newNode;
    newNode = (struct BTreeNode
*)malloc(sizeof(struct BTreeNode));
    newNode->val[1] = val;
    newNode->count = 1;
    newNode->link[0] = root;
    newNode->link[1] = child;
    return newNode;
}

void insertNode(int val, int pos, struct BTreeNode
*node,
               struct BTreeNode *child) {
    int j = node->count;
```

```

while (j > pos) {
    node->val[j + 1] = node->val[j];
    node->link[j + 1] = node->link[j];
    j--;
}
node->val[j + 1] = val;
node->link[j + 1] = child;
node->count++;
}

void splitNode(int val, int *pval, int pos, struct
BTreeNode *node,
               struct BTreeNode *child, struct BTreeNode
**newNode) {
    int median, j;

    if (pos > MIN)
        median = MIN + 1;
    else
        median = MIN;

    *newNode = (struct BTreeNode
*)malloc(sizeof(struct BTreeNode));
    j = median + 1;
    while (j <= MAX) {
        (*newNode)->val[j - median] = node->val[j];
        (*newNode)->link[j - median] = node->link[j];
        j++;
    }
    node->count = median;
    (*newNode)->count = MAX - median;

    if (pos <= MIN) {
        insertNode(val, pos, node, child);
    } else {
        insertNode(val, pos - median, *newNode, child);
    }
}

```

```

    }
    *pval = node->val[node->count];
    (*newNode)->link[0] = node->link[node->count];
    node->count--;
}

int setValue(int val, int *pval,
             struct BTreeNode *node, struct BTreeNode
**child) {
    int pos;
    if (!node) {
        *pval = val;
        *child = NULL;
        return 1;
    }

    if (val < node->val[1]) {
        pos = 0;
    } else {
        for (pos = node->count;
            (val < node->val[pos] && pos > 1); pos--)
            ;
        if (val == node->val[pos]) {
            printf("Duplicates are not permitted\n");
            return 0;
        }
    }

    if (setValue(val, pval, node->link[pos], child))
    {
        if (node->count < MAX) {
            insertNode(*pval, pos, node, *child);
        } else {
            splitNode(*pval, pval, pos, node, *child,
child);
            return 1;
        }
    }
}

```

```

    }
    return 0;
}

void insert(int val) {
    int flag, i;
    struct BTreeNode *child;

    flag = setValue(val, &i, root, &child);
    if (flag)
        root = createNode(i, child);
}

void search(int val, int *pos, struct BTreeNode
*myNode) {
    if (!myNode) {
        return;
    }

    if (val < myNode->val[1]) {
        *pos = 0;
    } else {
        for (*pos = myNode->count;
            (val < myNode->val[*pos] && *pos > 1);
            (*pos)--);
        ;
        if (val == myNode->val[*pos]) {
            printf("%d is found", val);
            return;
        }
    }
    search(val, pos, myNode->link[*pos]);

    return;
}

```

```

void traversal(struct BTreeNode *myNode) {
    int i;
    if (myNode) {
        for (i = 0; i < myNode->count; i++) {
            traversal(myNode->link[i]);
            printf("%d ", myNode->val[i + 1]);
        }
        traversal(myNode->link[i]);
    }
}

int main() {
    int val, ch;

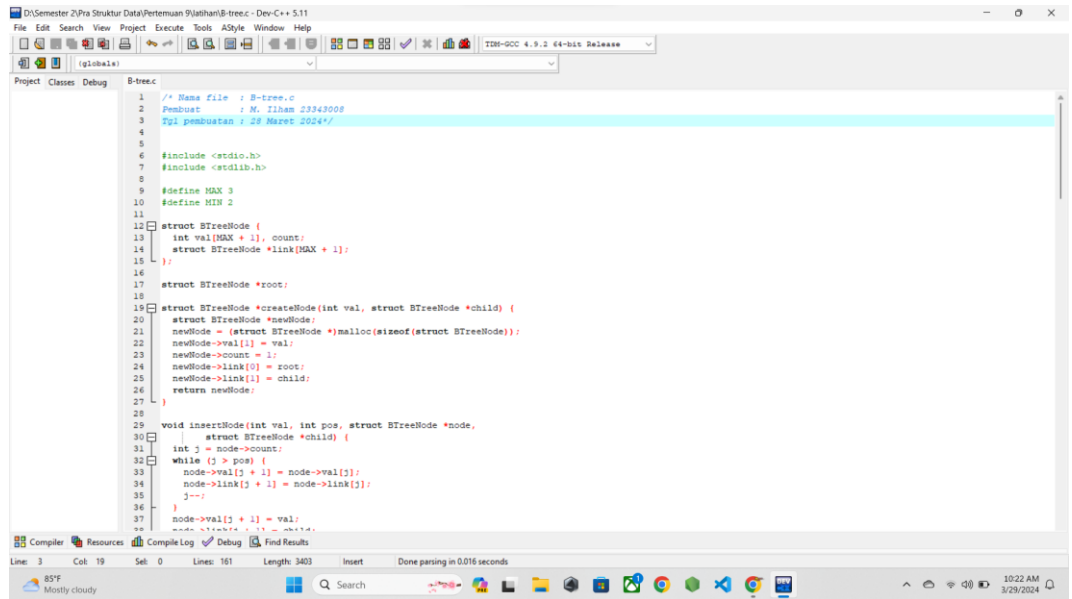
    insert(8);
    insert(9);
    insert(10);
    insert(11);
    insert(15);
    insert(16);
    insert(17);
    insert(18);
    insert(20);
    insert(23);

    traversal(root);

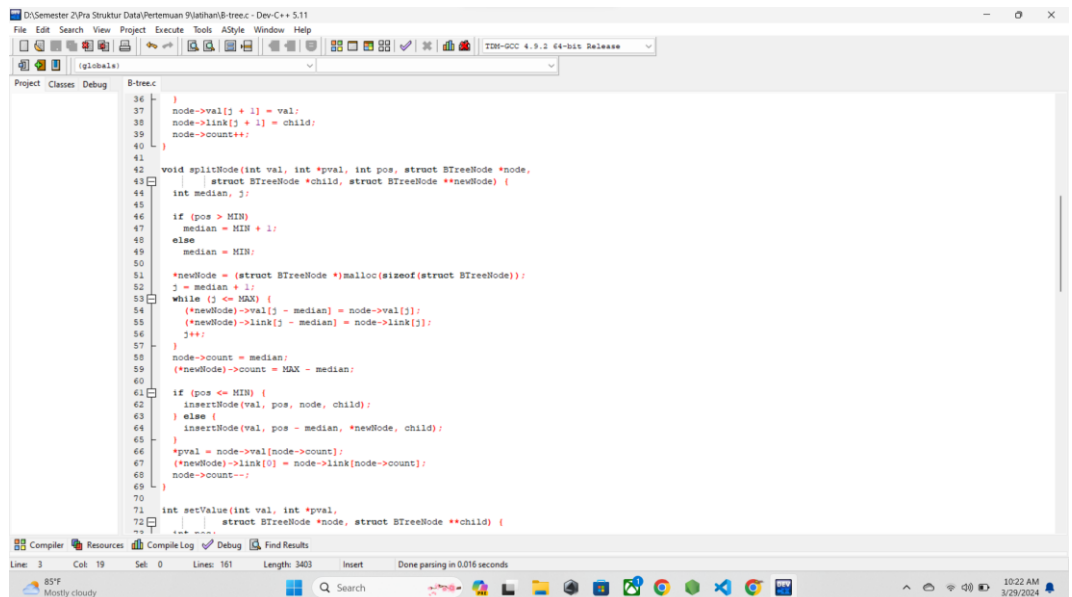
    printf("\n");
    search(11, &ch, root);
}

```

b. SCREENSHOT PROGRAM



```
1  /* Nama file : B-tree.c
2  Pendont : M. Ilham 23343008
3  Tgl penkutan : 28 Maret 2024 */
4
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  #define MAX 3
10 #define MIN 2
11
12 struct BTreeNode {
13     int val[MAX + 1]; count;
14     struct BTreeNode *link[MAX + 1];
15 };
16
17 struct BTreeNode *root;
18
19 struct BTreeNode *createNode(int val, struct BTreeNode *child) {
20     struct BTreeNode *newNode;
21     newNode = (struct BTreeNode *) malloc(sizeof(struct BTreeNode));
22     newNode->val[0] = val;
23     newNode->count = 1;
24     newNode->link[0] = root;
25     newNode->link[1] = child;
26     return newNode;
27 }
28
29 void insertNode(int val, int pos, struct BTreeNode *node,
30               struct BTreeNode *child) {
31     int i = node->count;
32     while (i > pos) {
33         node->val[i + 1] = node->val[i];
34         node->link[i + 1] = node->link[i];
35         i--;
36     }
37     node->val[i + 1] = val;
38     node->link[i + 1] = child;
39     node->count++;
40 }
```



```
36 }
37 node->val[i + 1] = val;
38 node->link[i + 1] = child;
39 node->count++;
40 }
41
42 void splitNode(int val, int *pval, int pos, struct BTreeNode *node,
43               struct BTreeNode *child, struct BTreeNode **newNode) {
44     int median, j;
45     if (pos > MIN) {
46         median = MIN + 1;
47     } else {
48         median = MIN;
49     }
50     *newNode = (struct BTreeNode *) malloc(sizeof(struct BTreeNode));
51     j = median + 1;
52     while (j <= MAX) {
53         (*newNode->val[j - median]) = node->val[j];
54         (*newNode->link[j - median]) = node->link[j];
55         j++;
56     }
57     node->count = median;
58     (*newNode->count) = MAX - median;
59
60     if (pos <= MIN) {
61         insertNode(val, pos, node, child);
62     } else {
63         insertNode(val, pos - median, *newNode, child);
64     }
65     *pval = node->val[node->count];
66     (*newNode->link[0]) = node->link[node->count];
67     node->count--;
68 }
69
70 int setValue(int val, int *pval,
71             struct BTreeNode *node, struct BTreeNode **child) {
72     // ...
73 }
```


1) createNode

Merupakan fungsi yang membuat node baru yang langsung menginisialisasi nilai pertama pada key nya dan menunjuk root serta child.

2) insertNode

Merupakan fungsi yang memasukkan value ke key pada node pada posisi yang ditentukan.

3) splitNode

Merupakan fungsi yang memecah node menjadi beberapa node saat nilai baru di inputkan pada satu node sudah memiliki value sebanyak batas maksimal.

SplitNode ini memecah node berdasarkan nilai tengahnya, media akan di jadikan sebagai parent serta nilai yang lebih kecil dan lebih besar akan dijadikan di left atau right sebagai child child nya.

4) setValue

Merupakan fungsi yang mengembalikan nilai 1 atau 0 atau melakukan rekursif hingga mendapatkan kondisi yang diinginkan. Misalnya saat dilakukan pada awal program maka akan mengembalikan 1 dan membuat node yang baru di buat menjadi root, sedangkan jika root sudah ada maka akan menginsert value di posisi tergantung value baru lebih kecil atau lebih besar dari value yang sudah ada pada node.

5) Insert

Merupakan fungsi untuk input value dan memanggil fungsi untuk createNode.

6) Search

Merupakan fungsi untuk mencari value tertentu dari tiap key pada node berdasarkan perbandingan value yang akan di cari dengan value yang sudah tersimpan pada node.

7) Traversal

Merupakan fungsi untuk menampilkan setiap nilai yang tersimpan pada key dari setiap node dalam B-Tree.

Pada program saya diatas, saya menginputkan 10 buah value yang masing masing akan disimpan pada setiap key pada node dalam B-Tree. Proses nya, pada awalnya akan di panggil fungsi insert yang pertama, lalu akan di panggil fungsi setValue lalu akan mengembalikan 1 dan di panggillah fungsi createNode yang akan mengembalikan node ke variabel dalam funggsi insert dan memasukkannya ke dalam variabel root.

Untuk insert selanjutnya selama ukurannya node masih belum melebihi batas maksimal, maka saat dalam proses fungsi insert akan mengembalikan 1 juga karena proses rekursif pemanggilan fungsi setValue selama node yang dijadikan argument belum NULL. Jika sudah null maka akan mengembalikan 1 dan menjalankan blok if terakhir pada setValue yang akan menjalankan insertNode().

Selanjutnya pada insert yang sudah memiliki key pada node yang jumlahnya melebihi batas maksimal, maka prosesnya sama seperti sebelumnya, bedanya akan di lakukan splitNode() alih alih insertNode() untuk memecah nodenya terlebih dahulu barulah akan dijalankan fungsi insertNode() setelah nodenya di pecah.

Lalu setelah di inputkan 10 buah value, maka akan di cari value dengan nilai 11 yang akan menjalankan fungsi search dan mencari ke setiap key dari node yang memiliki nilai sama dengan nilai yang akan di cari. Jika ditemukan maka akan mengembalikan string bahwa value ditemukan, jika tidak maka tidak akan menghasilkan apa apa.

2. tugas-B-tree.c

a. SOURCE CODE

```
/* Nama file   : tugas-B-tree.c
Pembuat       : M. Ilham 23343008
Tgl pembuatan : 28 Maret 2024*/

#include <stdio.h>
#include <stdlib.h>

#define MAX 3
#define MIN 2

struct BTreeNode {
    int val[MAX + 1], count;
    struct BTreeNode *link[MAX + 1];
};

struct BTreeNode *root;

struct BTreeNode *createNode(int val, struct
BTreeNode *child) {
    struct BTreeNode *newNode;
    newNode = (struct BTreeNode
*)malloc(sizeof(struct BTreeNode));
    newNode->val[1] = val;
    newNode->count = 1;
    newNode->link[0] = root;
    newNode->link[1] = child;
    return newNode;
}

void insertNode(int val, int pos, struct BTreeNode
*node,
                struct BTreeNode *child) {
    int j = node->count;
```

```

while (j > pos) {
    node->val[j + 1] = node->val[j];
    node->link[j + 1] = node->link[j];
    j--;
}
node->val[j + 1] = val;
node->link[j + 1] = child;
node->count++;
}

void splitNode(int val, int *pval, int pos, struct
BTreeNode *node,
               struct BTreeNode *child, struct BTreeNode
**newNode) {
    int median, j;

    if (pos > MIN)
        median = MIN + 1;
    else
        median = MIN;

    *newNode = (struct BTreeNode
*)malloc(sizeof(struct BTreeNode));
    j = median + 1;
    while (j <= MAX) {
        (*newNode)->val[j - median] = node->val[j];
        (*newNode)->link[j - median] = node->link[j];
        j++;
    }
    node->count = median;
    (*newNode)->count = MAX - median;

    if (pos <= MIN) {
        insertNode(val, pos, node, child);
    } else {
        insertNode(val, pos - median, *newNode, child);
    }
}

```

```

    }
    *pval = node->val[node->count];
    (*newNode)->link[0] = node->link[node->count];
    node->count--;
}

int setValue(int val, int *pval,
             struct BTreeNode *node, struct BTreeNode
**child) {
    int pos;
    if (!node) {
        *pval = val;
        *child = NULL;
        return 1;
    }

    if (val < node->val[1]) {
        pos = 0;
    } else {
        for (pos = node->count;
            (val < node->val[pos] && pos > 1); pos--)
            ;
        if (val == node->val[pos]) {
            printf("Duplicates are not permitted\n");
            return 0;
        }
    }

    if (setValue(val, pval, node->link[pos], child))
    {
        if (node->count < MAX) {
            insertNode(*pval, pos, node, *child);
        } else {
            splitNode(*pval, pval, pos, node, *child,
child);
            return 1;
        }
    }
}

```

```

    }
    return 0;
}

void insert(int val) {
    int flag, i;
    struct BTreeNode *child;

    flag = setValue(val, &i, root, &child);
    if (flag)
        root = createNode(i, child);
}

void search(int val, int *pos, struct BTreeNode
*myNode) {
    if (!myNode) {
        return;
    }

    if (val < myNode->val[1]) {
        *pos = 0;
    } else {
        for (*pos = myNode->count;
            (val < myNode->val[*pos] && *pos > 1);
            (*pos)--);
        ;
        if (val == myNode->val[*pos]) {
            printf("%d is found", val);
            return;
        }
        else {
            printf("%d is not found", val);
            return;
        }
    }
    search(val, pos, myNode->link[*pos]);
}

```

```

        return;
    }

void traversal(struct BTreeNode *myNode) {
    int i;
    if (myNode) {
        for (i = 0; i < myNode->count; i++) {
            traversal(myNode->link[i]);
            printf("%d ", myNode->val[i + 1]);
        }
        traversal(myNode->link[i]);
    }
}

int main() {
    int val, ch;

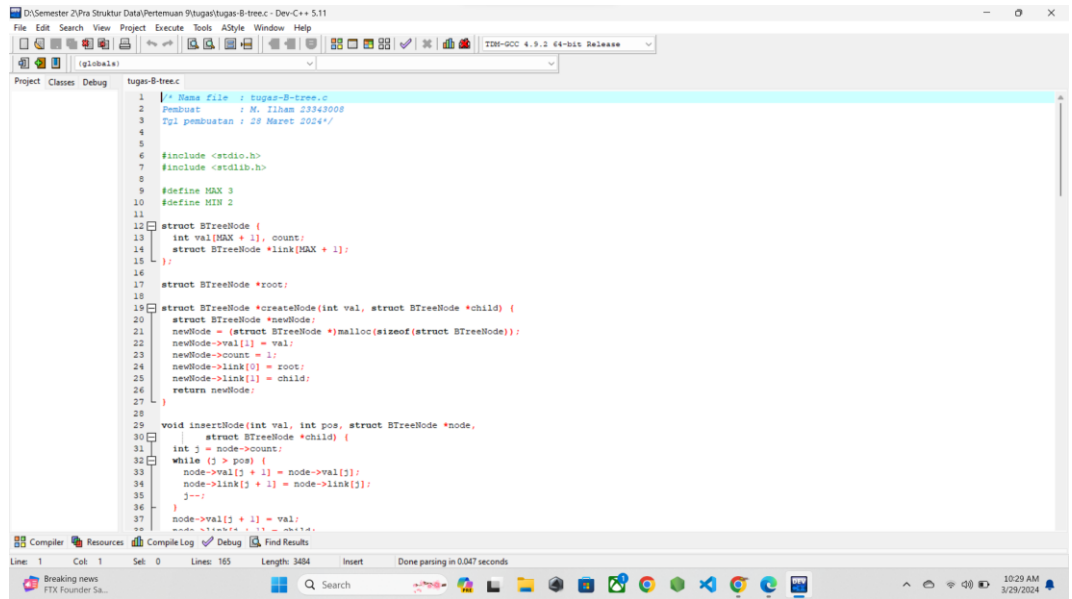
    insert(20);
    insert(25);
    insert(30);
    insert(33);
    insert(34);
    insert(40);
    insert(50);
    insert(55);
    insert(60);
    insert(61);

    traversal(root);

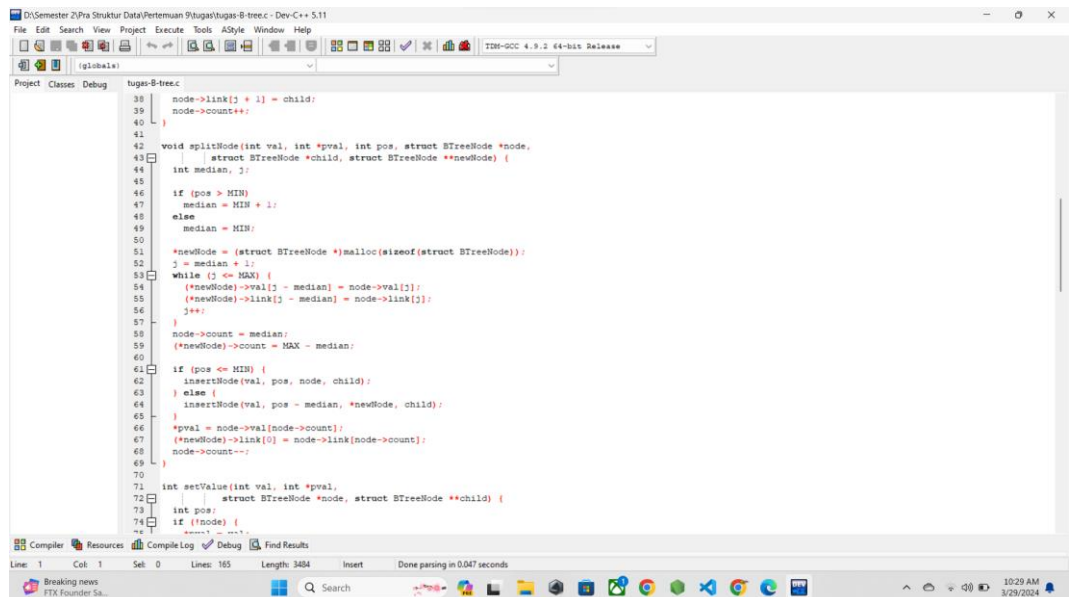
    printf("\n");
    search(22, &ch, root);
}

```


b. SCREENSHOT PROGRAM



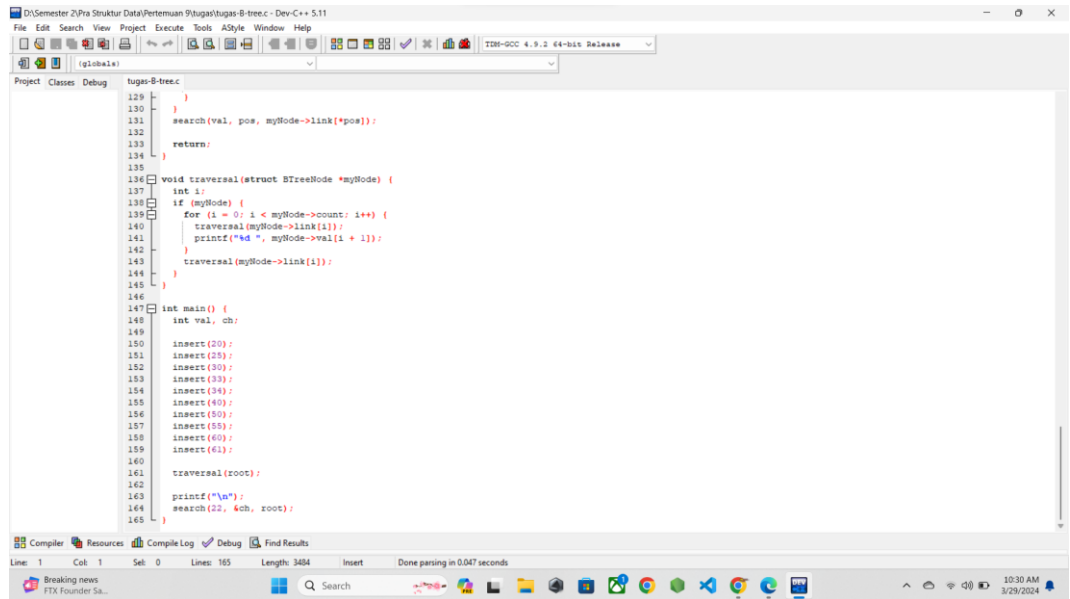
```
1  /* Nama file : tugas-B-tree.c
2  Pembuat : M. Ilham 22342008
3  Tgl pembuatan : 28 Maret 2024 */
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  #define MAX 3
9  #define MIN 2
10
11  struct BTreeNode {
12      int val[MAX + 1], count;
13      struct BTreeNode *link[MAX + 1];
14  };
15
16  struct BTreeNode *root;
17
18  struct BTreeNode *createNode(int val, struct BTreeNode *child) {
19      struct BTreeNode *newNode;
20      newNode = (struct BTreeNode *)malloc(sizeof(struct BTreeNode));
21      newNode->val[] = val;
22      newNode->count = 1;
23      newNode->link[0] = root;
24      newNode->link[1] = child;
25      return newNode;
26  }
27
28  void insertNode(int val, int pos, struct BTreeNode *node,
29                  struct BTreeNode *child) {
30      int i = node->count;
31      while (i > pos) {
32          node->val[i + 1] = node->val[i];
33          node->link[i + 1] = node->link[i];
34          i--;
35      }
36      node->val[i + 1] = val;
37      node->link[i + 1] = child;
38      node->count++;
39  }
```



```
38      node->link[i + 1] = child;
39      node->count++;
40  }
41
42  void splitNode(int val, int *pval, int pos, struct BTreeNode *node,
43                struct BTreeNode *child, struct BTreeNode **newNode) {
44      int median, j;
45
46      if (pos > MIN)
47          median = MIN + 1;
48      else
49          median = MIN;
50
51      *newNode = (struct BTreeNode *)malloc(sizeof(struct BTreeNode));
52      j = median + 1;
53      while (j <= MAX) {
54          (*newNode)->val[j - median] = node->val[j];
55          (*newNode)->link[j - median] = node->link[j];
56          j++;
57      }
58      node->count = median;
59      (*newNode)->count = MAX - median;
60
61      if (pos <= MIN) {
62          insertNode(val, pos, node, child);
63      } else {
64          insertNode(val, pos - median, *newNode, child);
65      }
66      *pval = node->val[node->count];
67      (*newNode)->link[0] = node->link[node->count];
68      node->count--;
69  }
70
71  int setValue(int val, int *pval,
72              struct BTreeNode *node, struct BTreeNode **child) {
73      int pos;
74      if (!node) {
75          return -1;
76      }
77      if (node->count == 0) {
78          *pval = node->val[0];
79          *child = node->link[0];
80          return 0;
81      }
82      if (node->val[node->count - 1] < val) {
83          return 1;
84      }
85      if (node->val[0] > val) {
86          return 2;
87      }
88      return 0;
89  }
```

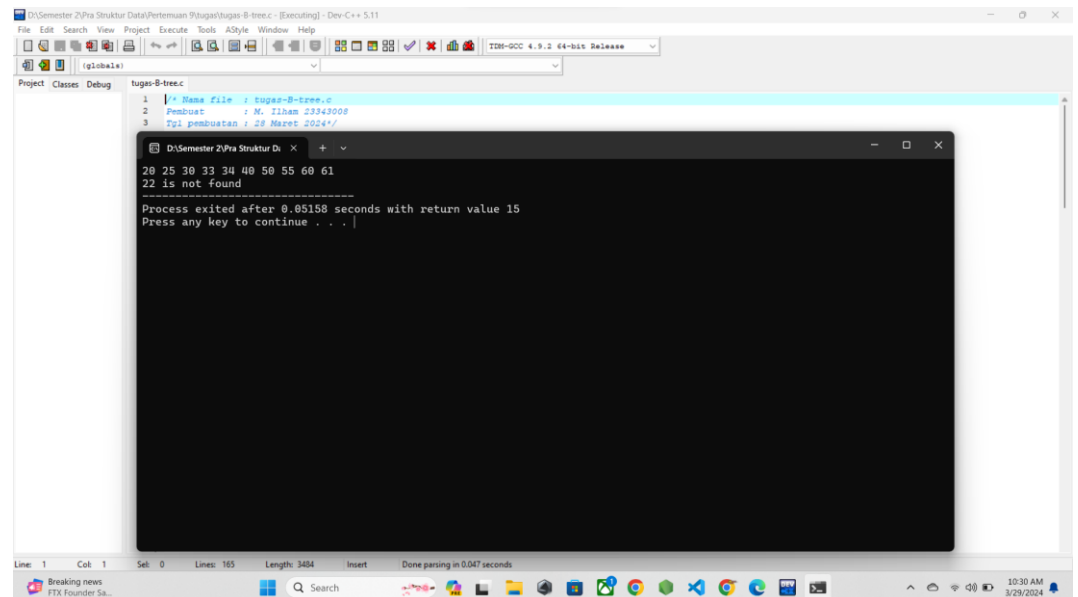
```
D:\Semester 2\Pro Struktur Data\Peraturan 9\tugas\tugas-B-tree.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
[Icons] [Compiler] [Debugger] [Global] [TDM-GCC 4.9.2 64-bit Release]
Project Classes Debug tugas-B-tree.c
75 *pval = val;
76 *child = NULL;
77 return 1;
78 }
79
80 if (val < node->val[1]) {
81 pos = 0;
82 } else {
83 for (pos = node->count;
84 (val < node->val[pos] && pos > 1); pos--)
85 ;
86 if (val == node->val[pos]) {
87 printf("Duplicates are not permitted\n");
88 return 0;
89 }
90 }
91 if (setValue(val, pval, node->link[pos], child)) {
92 if (node->count < MAX) {
93 insertNode(*pval, pos, node, *child);
94 } else {
95 splitNode(*pval, pval, pos, node, *child, child);
96 return 1;
97 }
98 }
99 return 0;
100 }
101
102 void insert(int val) {
103 int flag, i;
104 struct BTreeNode *child;
105
106 flag = setValue(val, &i, root, &child);
107 if (flag)
108 root = createNode(i, child);
109 }
110
111 void search(int val, int *pos, struct BTreeNode *myNode) {
112 if (!myNode) {
113 return;
114 }
115 if (val < myNode->val[1]) {
116 *pos = 0;
117 } else {
118 for (*pos = myNode->count;
119 (val < myNode->val[*pos] && *pos > 1); (*pos)--);
120 if (val == myNode->val[*pos]) {
121 printf("kd is found", val);
122 return;
123 }
124 else {
125 printf("kd is not found", val);
126 return;
127 }
128 search(val, pos, myNode->link[*pos]);
129 return;
130 }
131
132 void traversal(struct BTreeNode *myNode) {
133 int i;
134 if (myNode) {
135 for (i = 0; i < myNode->count; i++) {
136 traversal(myNode->link[i]);
137 printf("%d ", myNode->val[i + 1]);
138 }
139 traversal(myNode->link[i]);
140 }
141 }
142
143 int main() {
144 int val, ch;
145 ...
146 }
```

```
D:\Semester 2\Pro Struktur Data\Peraturan 9\tugas\tugas-B-tree.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
[Icons] [Compiler] [Debugger] [Global] [TDM-GCC 4.9.2 64-bit Release]
Project Classes Debug tugas-B-tree.c
112 if (!myNode) {
113 return;
114 }
115 if (val < myNode->val[1]) {
116 *pos = 0;
117 } else {
118 for (*pos = myNode->count;
119 (val < myNode->val[*pos] && *pos > 1); (*pos)--);
120 if (val == myNode->val[*pos]) {
121 printf("kd is found", val);
122 return;
123 }
124 else {
125 printf("kd is not found", val);
126 return;
127 }
128 search(val, pos, myNode->link[*pos]);
129 return;
130 }
131
132 void traversal(struct BTreeNode *myNode) {
133 int i;
134 if (myNode) {
135 for (i = 0; i < myNode->count; i++) {
136 traversal(myNode->link[i]);
137 printf("%d ", myNode->val[i + 1]);
138 }
139 traversal(myNode->link[i]);
140 }
141 }
142
143 int main() {
144 int val, ch;
145 ...
146 }
```



```
129 }
130 }
131 search(val, pos, myNode->link[pos]);
132
133 return;
134 }
135
136 void traversal(struct BTreeNode *myNode) {
137     int i;
138     if (myNode) {
139         for (i = 0; i < myNode->count; i++) {
140             traversal(myNode->link[i]);
141             printf("%d ", myNode->val[i + 1]);
142         }
143         traversal(myNode->link[i]);
144     }
145 }
146
147 int main() {
148     int val, ch;
149
150     insert(20);
151     insert(25);
152     insert(30);
153     insert(33);
154     insert(34);
155     insert(40);
156     insert(50);
157     insert(55);
158     insert(60);
159     insert(61);
160
161     traversal(root);
162
163     printf("\n");
164     search(22, &ch, root);
165 }
```

c. SCREENSHOT OUTPUT



```
1 // Main file : tugas-B-tree.c
2 Pembuat : M. Iham 23343008
3 Tgl pembuatan : 28 Maret 2024//

20 25 30 33 34 40 50 55 60 61
22 is not found
-----
Process exited after 0.05158 seconds with return value 15
Press any key to continue . . .
```

d. PENJELASAN PROGRAM

Program diatas adalah program implementasi searching dengan menggunakan B-Tree pada modul multiway search tree. Namun pada program ini sedikit di modifikasi sehingga saat melakukan pencarian pada value yang tidak terdapat pada semua node dalam program maka akan menghasilkan string value tidak ditemukan alih alih tidak melakukan apapun. Pada program, terdapat beberapa algoritma untuk membuat B-Tree berjalan dengan lancar seperti sebelumnya yaitu 7 buah fungsi sebagai berikut :

createNode, insertNode, splitNode, setValue, insert, search, traversal.

Berikut penjelasan seetiap fungsinya :

1) createNode

Merupakan fungsi yang membuat node baru yang langsung menginisialisasi nilai pertama pada key nya dan menunjuk root serta child.

2) insertNode

Merupakan fungsi yang memasukkan value ke key pada node pada posisi yang ditentukan.

3) splitNode

Merupakan fungsi yang memecah node menjadi beberapa node saat nilai baru di inputkan pada satu node sudah memiliki value sebanyak batas maksimal.

SplitNode ini memecah node berdasarkan nilai tengahnya, media akan di jadikan sebagai parent serta nilai yang lebih kecil dan lebih besar akan dijadikan di left atau right sebagai child child nya.

4) setValue

Merupakan fungsi yang mengembalikan nilai 1 atau 0 atau melakukan rekursif hingga mendapatkan kondisi yang diinginkan.

Misalnya saat dilakukan pada awal program maka akan mengembalikan 1 dan membuat node yang baru di buat menjadi root, sedangkan jika root sudah ada maka akan menginsert value di posisi tergantung value baru lebih kecil atau lebih besar dari value yang sudah ada pada node.

5) Insert

Merupakan fungsi untuk input value dan memanggil fungsi untuk createNode.

6) Search

Merupakan fungsi untuk mencari value tertentu dari tiap key pada node berdasarkan perbandingan value yang akan di cari dengan value yang sudah tersimpan pada node.

7) Traversal

Merupakan fungsi untuk menampilkan setiap nilai yang tersimpan pada key dari setiap node dalam B-Tree.

Pada program saya diatas, saya juga menginputkan 10 buah value yang masing masing akan disimpan pada setiap key pada node dalam B-Tree. Untuk proses inputnya masih sama dengan program B-Tree sebelumnya.

Lalu setelah di inputkan 10 buah value, maka akan di cari value dengan nilai 11 yang akan menjalankan fungsi search dan mencari ke setiap key dari node yang memiliki nilai sama dengan nilai yang akan di cari. Jika ditemukan maka akan mengembalikan string bahwa value ditemukan, disini perbedaannya, pada program kali ini jika value tidak ditemukan maka akan menghasilkan output value tidak di temukan alih alih tidak menghasilkan apa apa.