**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**
   1. **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);

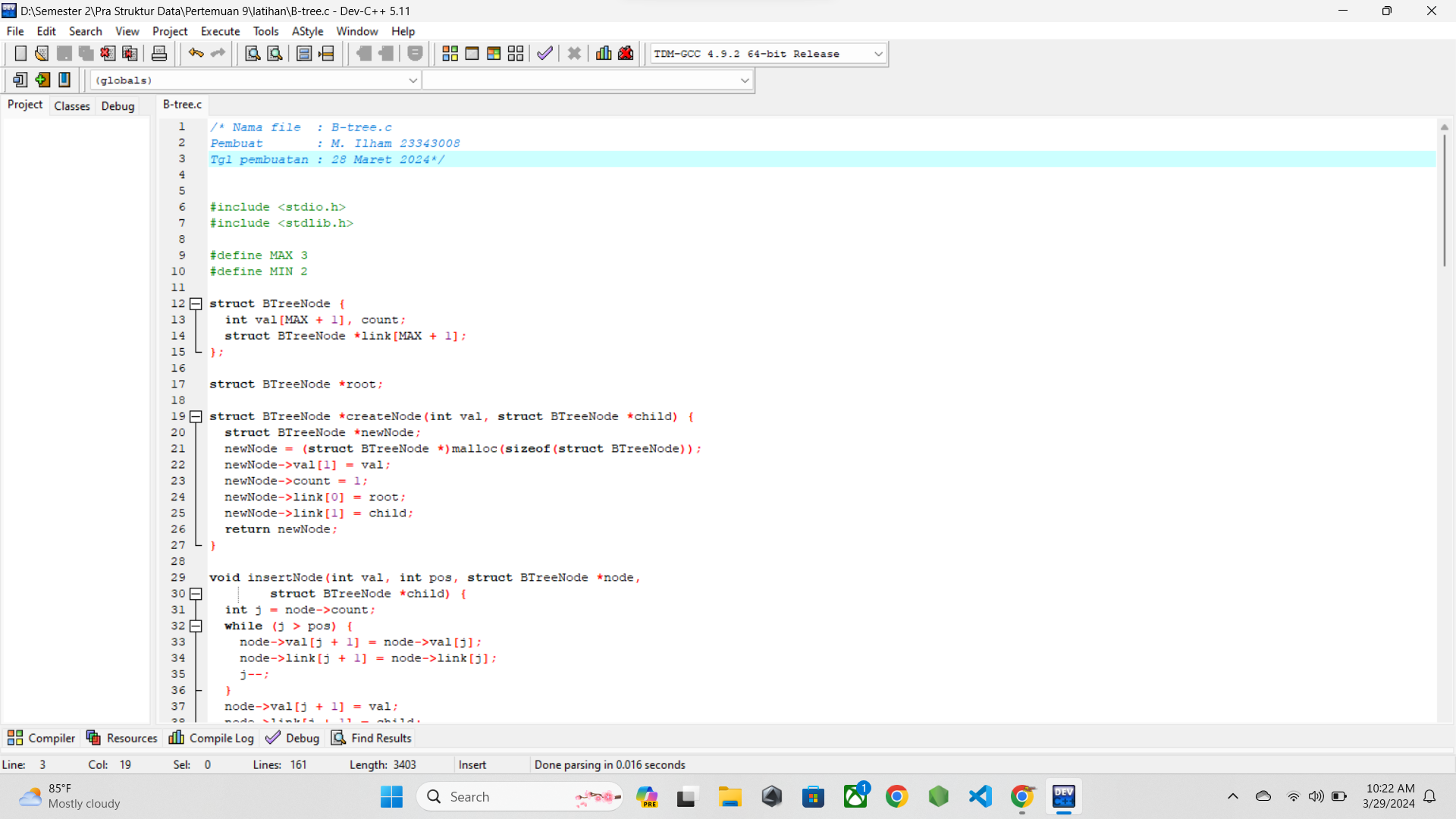
traversal(root);

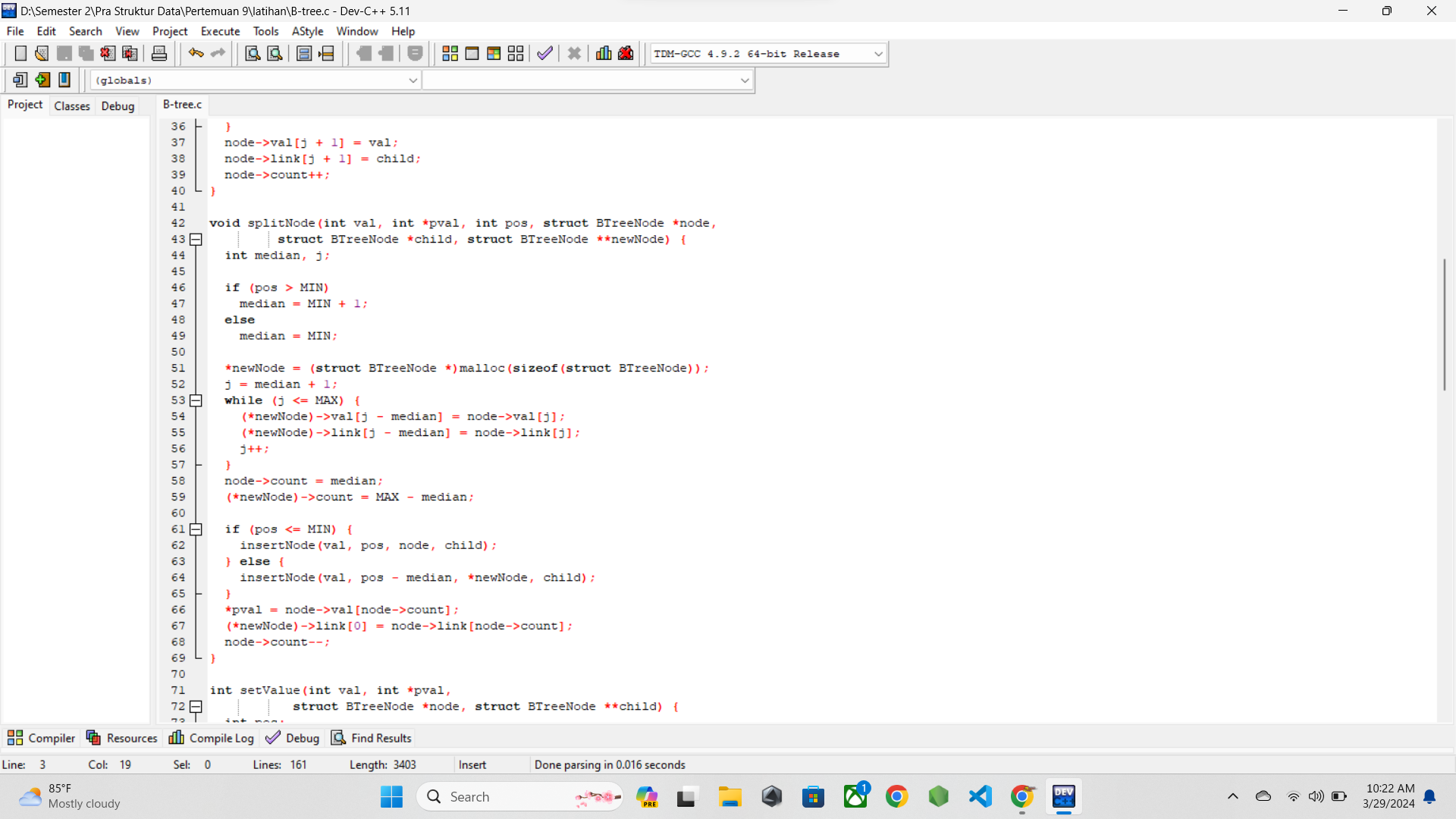
printf("\n");

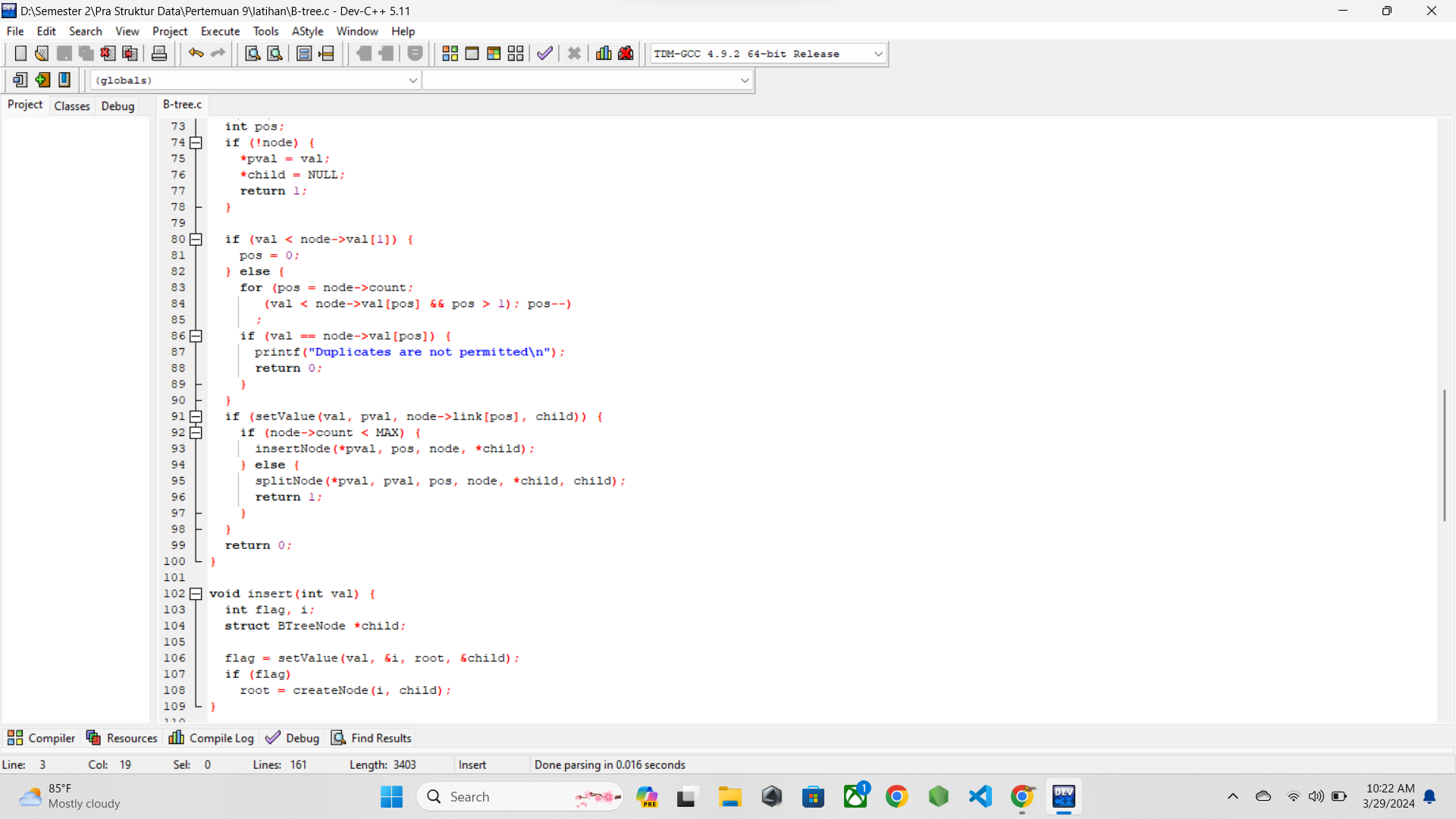
search(11, &ch, root);

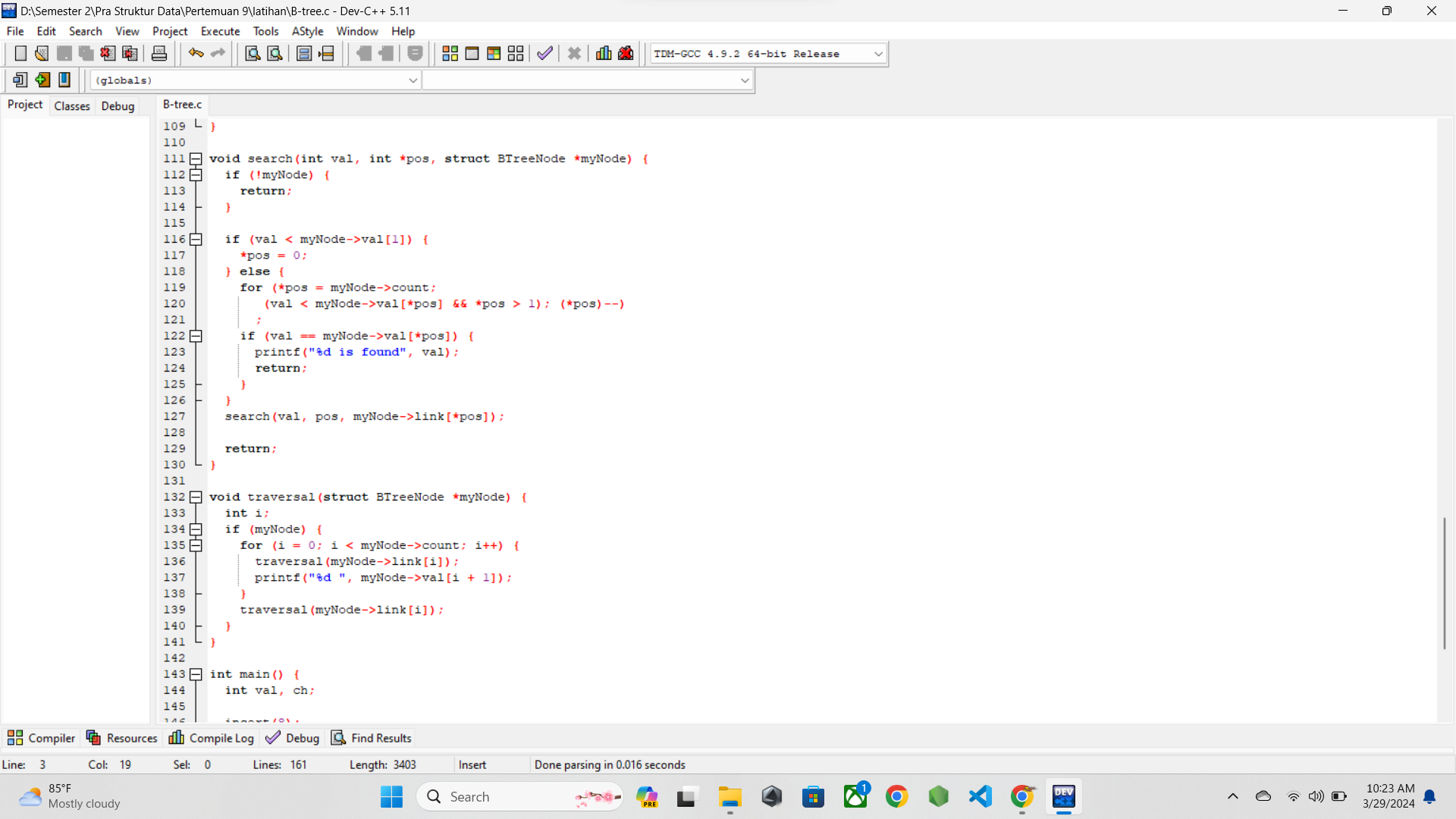
}

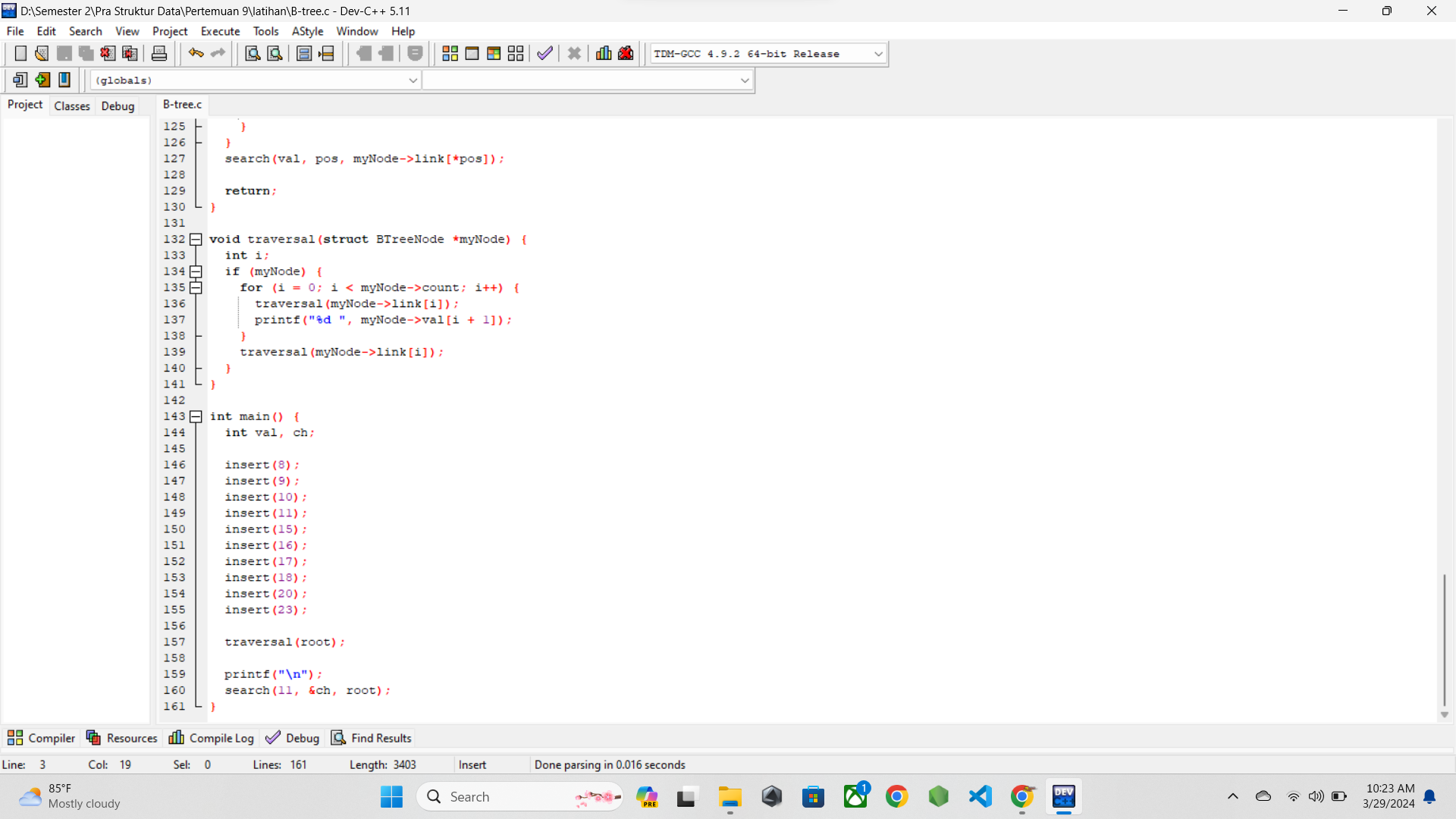
* 1. **SCREENSHOT PROGRAM**



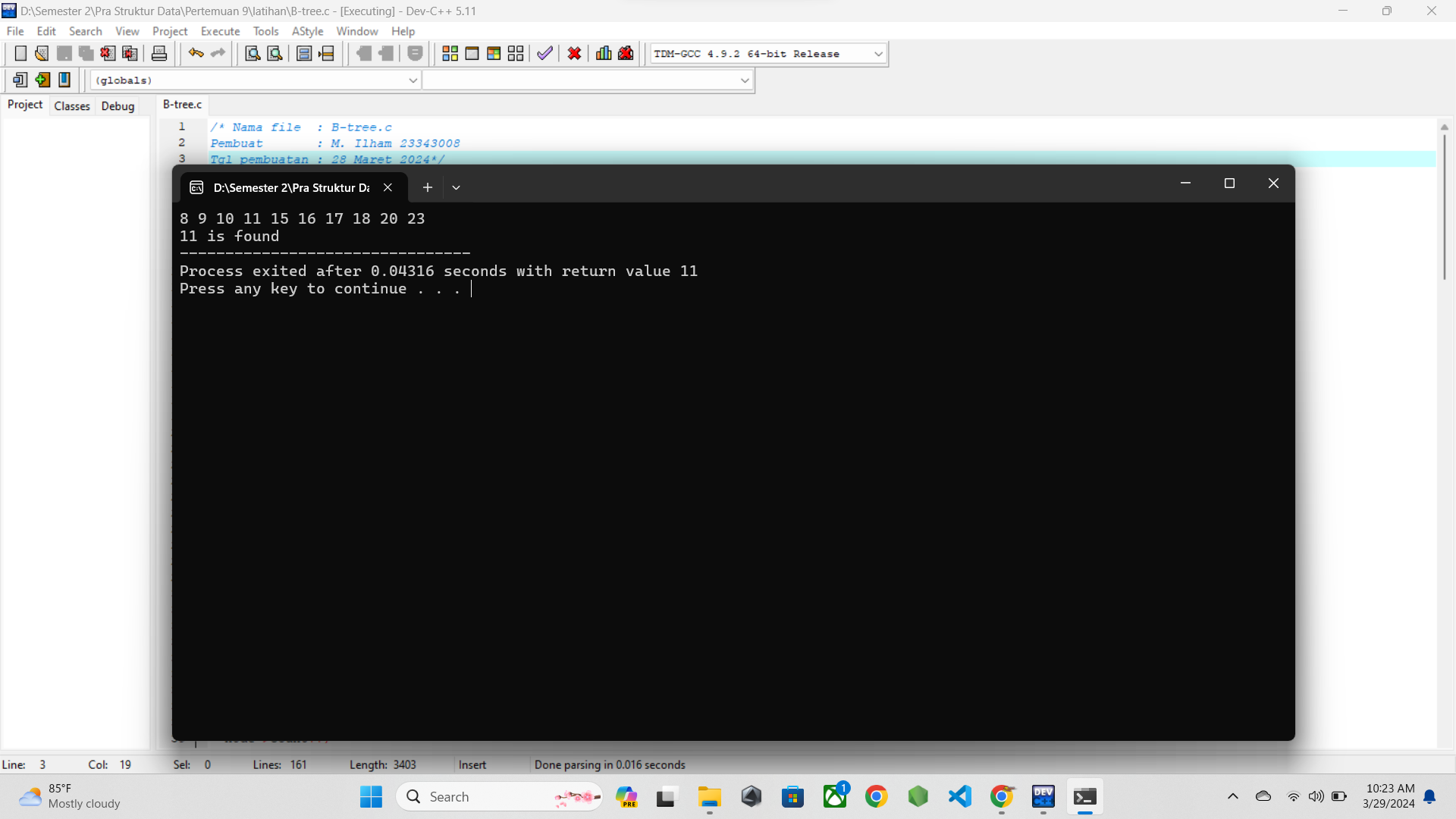








* 1. **SCREENSHOT OUTPUT**



* 1. **PENJELASAN PROGRAM**

Program diatas adalah program implementasi searching dengan menggunakan B-Tree pada modul multiway search tree. Pada program, terdapat beberapa algoritma untuk membuat B-Tree berjalan dengan lancer 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.

1. insertNode

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

1. 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.

1. 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.

1. Insert

Merupakan fungsi untuk input value dan memanggil fungsi untuk createNode.

1. 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.

1. 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 1juga 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.

1. **tugas-B-tree.c**
   1. **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);

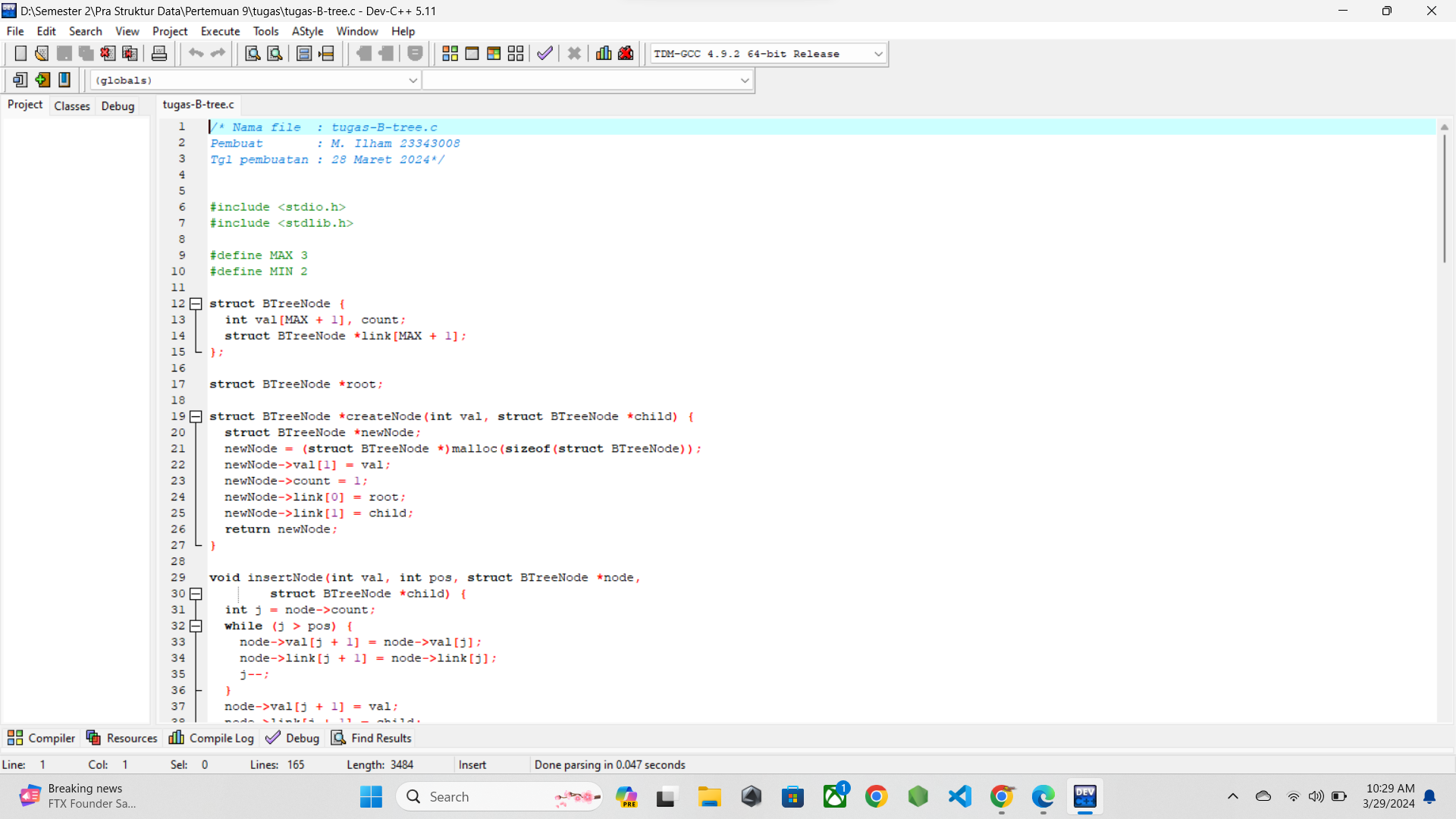
traversal(root);

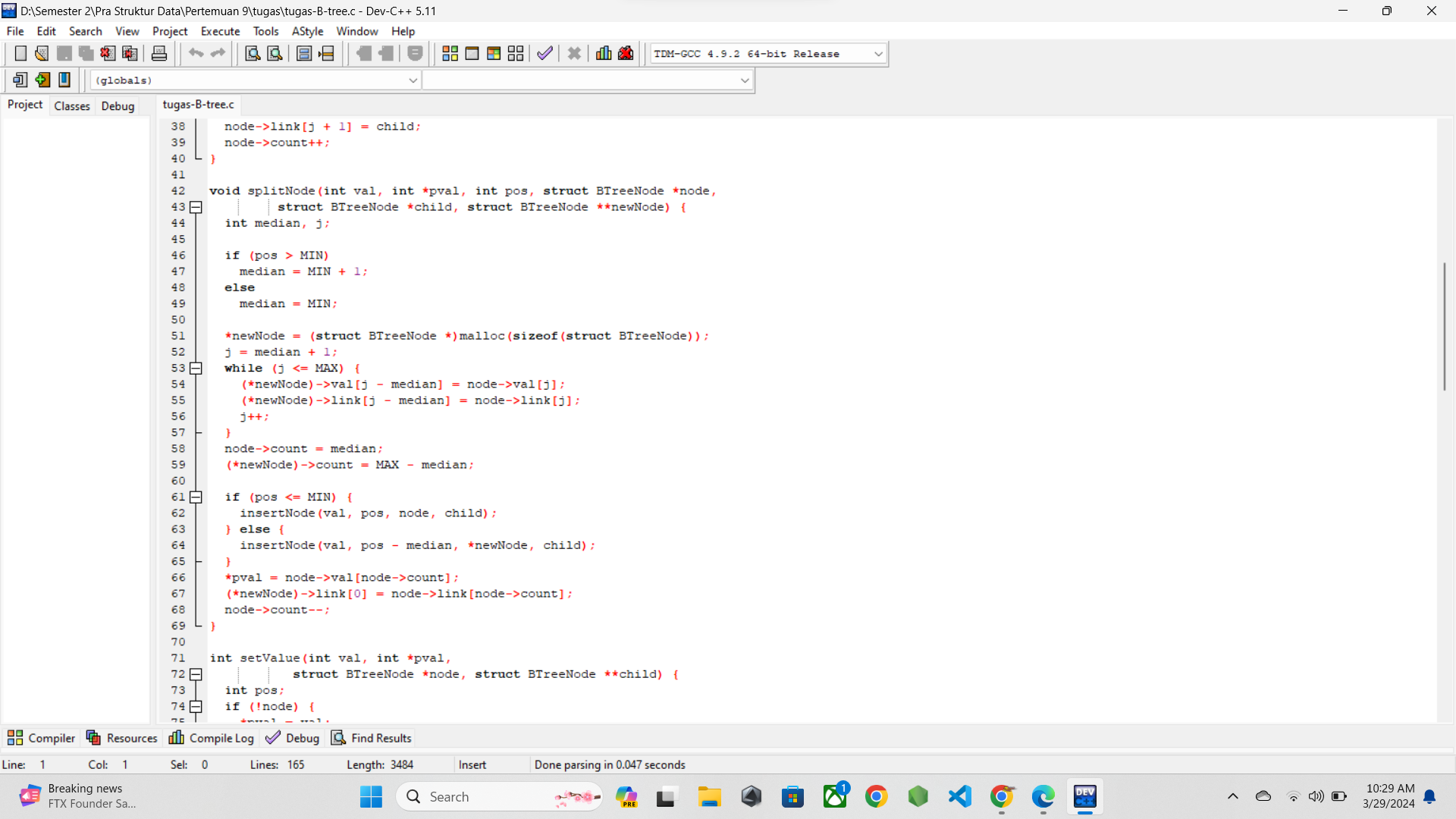
printf("\n");

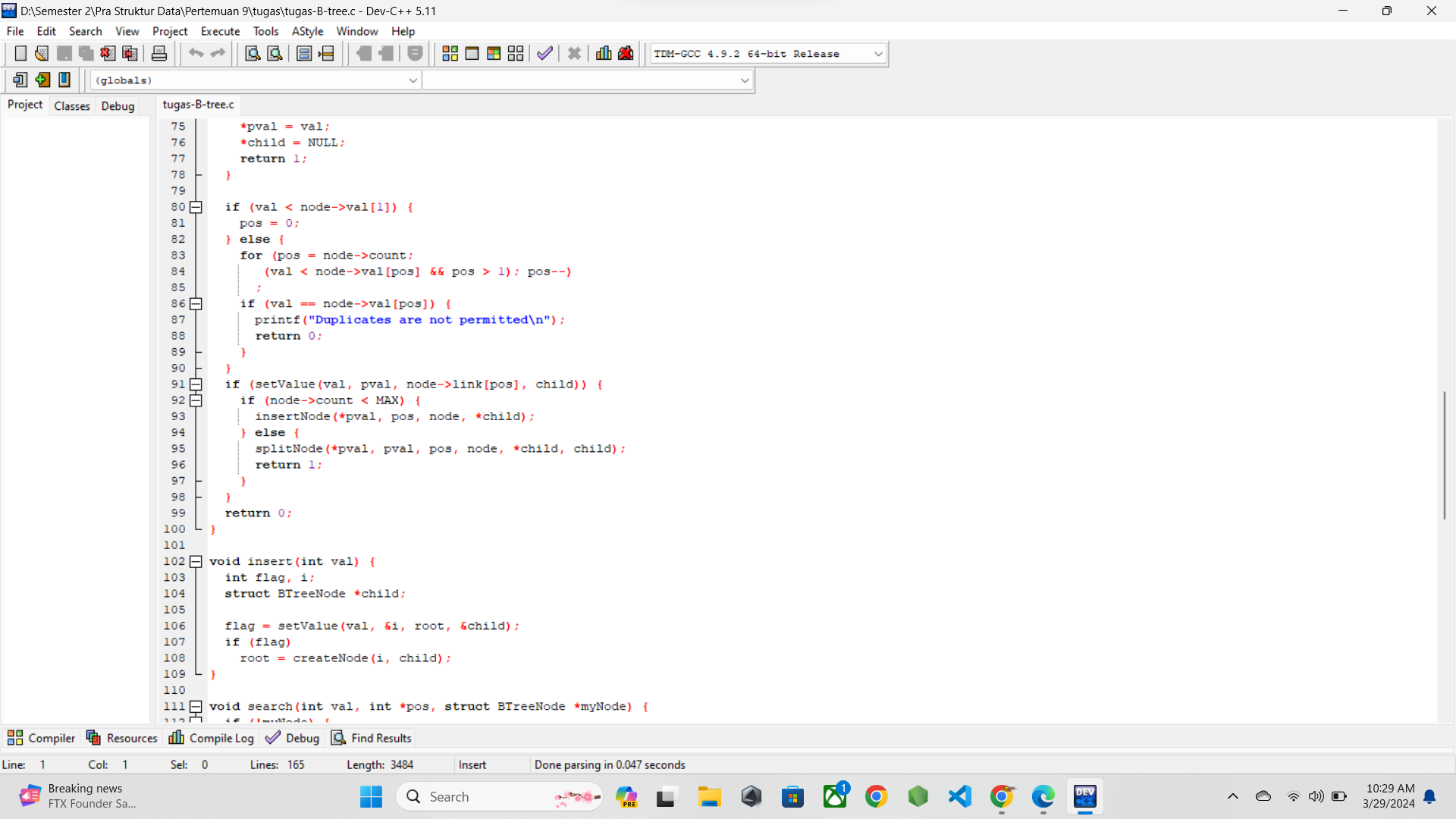
search(22, &ch, root);

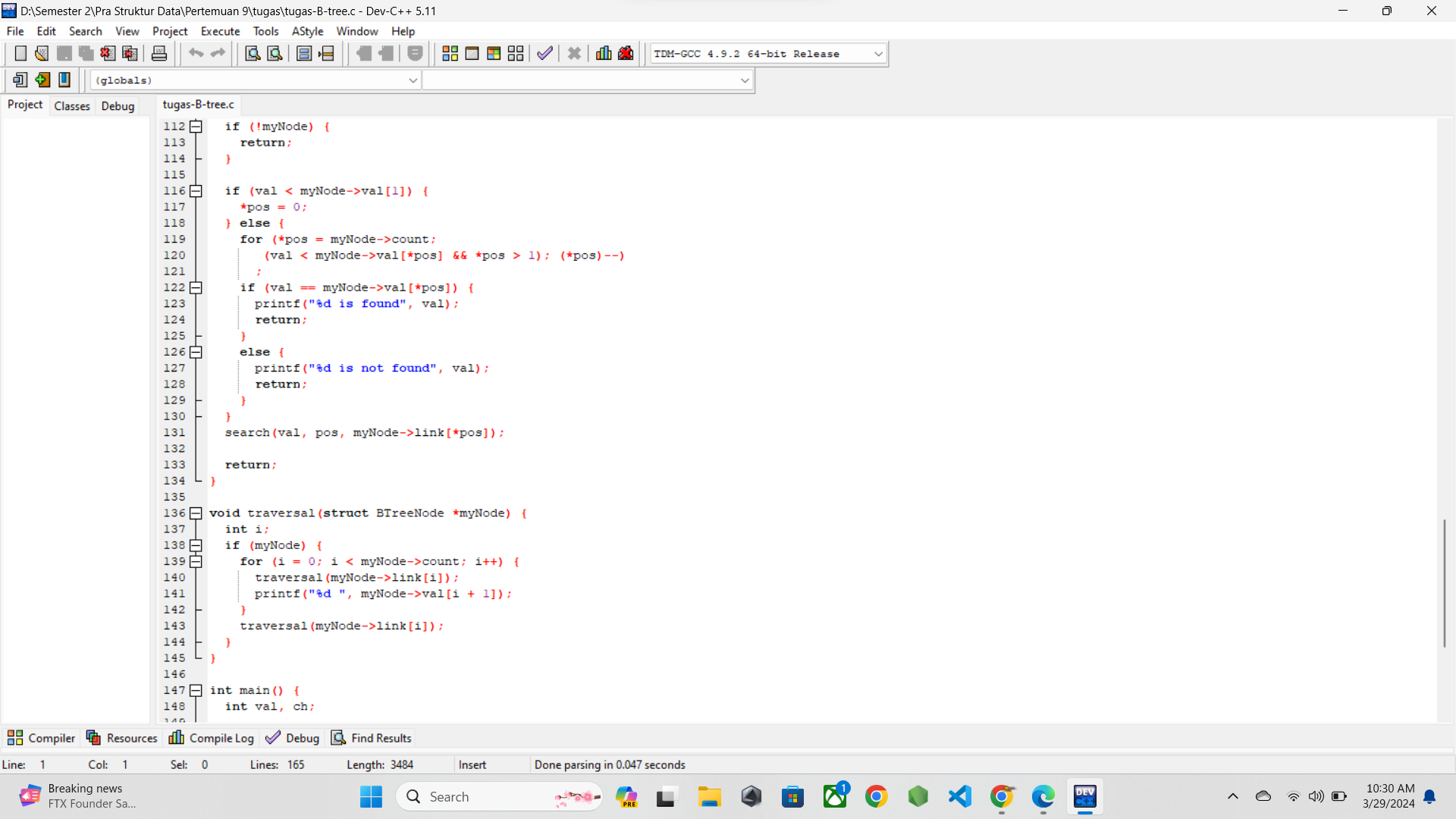
}

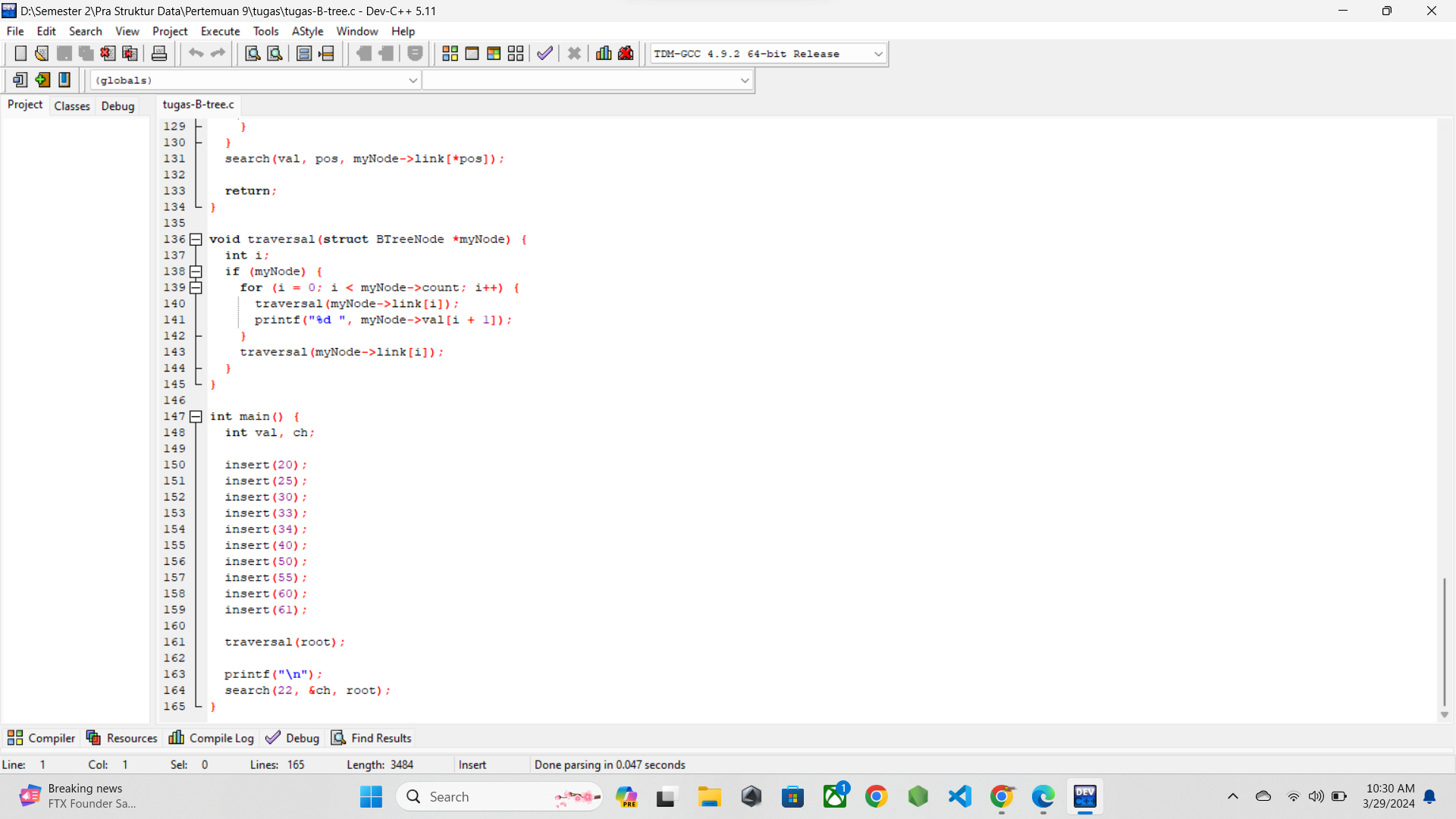
* 1. **SCREENSHOT PROGRAM**



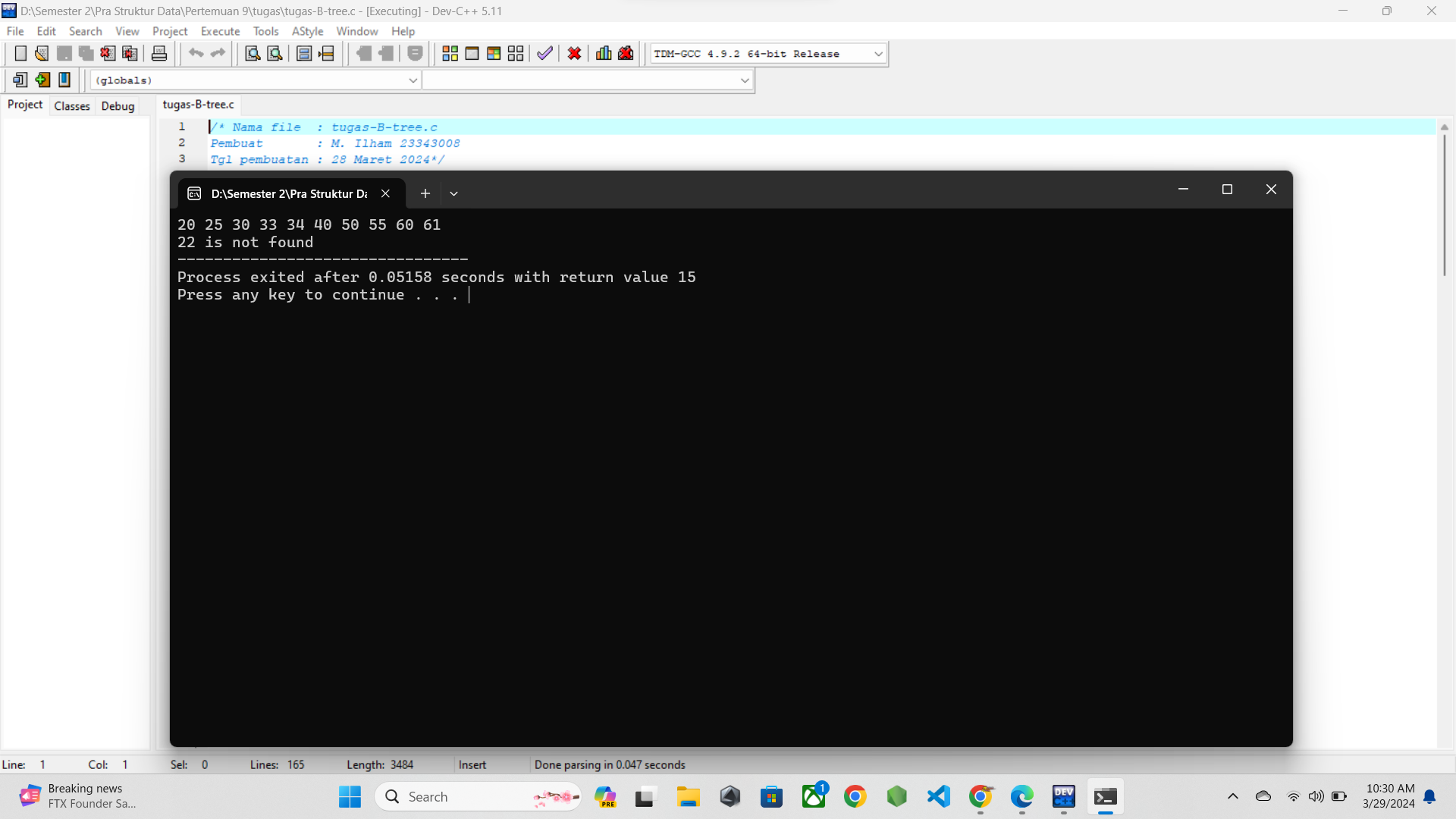








* 1. **SCREENSHOT OUTPUT**



* 1. **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 lancer 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.

1. insertNode

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

1. 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.

1. 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.

1. Insert

Merupakan fungsi untuk input value dan memanggil fungsi untuk createNode.

1. 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.

1. 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.