**Laporan Tugas Kecil 1**

**Penyelesaian *Word Search Puzzle* dengan Algoritma *Brute Force***

Mata Kuliah IF2211 - Strategi Algoritma



Oleh :

Nama : Jundan Haris

NIM : 13520155

Kelas : 02

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021/2022

# DAFTAR ISI

[DAFTAR ISI 1](#_Toc94076247)

[BAB I ALGORITMA *BRUTE FORCE* 2](#_Toc94076248)

[Algoritma *Brute Force* 2](#_Toc94076249)

[Algoritma *Brute Force* Untuk Menyelesaikan *Word Search Puzzle* 2](#_Toc94076250)

[BAB II *SOURCE PROGRAM* DALAM BAHASA C 3](#_Toc94076251)

[Modularity Program 3](#_Toc94076252)

[main.c 3](#_Toc94076253)

[reader.h 3](#_Toc94076254)

[reader.c 4](#_Toc94076255)

[matrix.h 5](#_Toc94076256)

[matrix.c 6](#_Toc94076257)

[charmachine.h 6](#_Toc94076258)

[charmachine.c 7](#_Toc94076259)

[BAB III CONTOH *INPUT* DAN *OUTPUT* PROGRAM 8](#_Toc94076260)

[BAB IV KESIMPULAN DAN SARAN 9](#_Toc94076261)

[Kesimpulan 9](#_Toc94076262)

[Saran 9](#_Toc94076263)

[REFERENSI 10](#_Toc94076264)

# BAB I ALGORITMA *BRUTE FORCE*

## Algoritma *Brute Force*

Algoritma *brute force* yaitu pendekatan yang lempang (*straightfordward*) untuk memecahkan suatu masalah atau persoalan. Algoritma ini biasanya didasarkan pada pernyataan pada persoalan atau definisi/konsep yang dilibatkan. Beberapa karakteristik algoritma *brute force* yaitu :

1. Algoritmanya tidak “cerdas” dan tidak mangkus. Membutuhkan volume komputasi yang besar dan waktu yang lama untuk menyelesaikan persoalan.
2. Algoritma ini lebih cocok untuk persoalan yang masukan atau n nya kecil.
3. Walaupun terdapat banyak kekuranga, algoritma ini hampir bisa memecahkan semua persoalan.

## Algoritma *Brute Force* Untuk Menyelesaikan *Word Search Puzzle*

Program yang dibuat menggunakan algoritma *brute force* untuk menyelesaikan persoalan, dalam hal ini yaitu mencari semua kata pada permainan *word search puzzle.* Langkah-langkah program secara garis besar yaitu :

1. Pertama-tama program akan membaca file puzzle yang akan diselesaikan lalu menyimpan puzzle dan daftar kata yang dicari.
2. Dari daftar kata yang ada, akan dicari satu persatu pada puzzle yang telah disimpan.
3. Kata yang akan dicari, diambil huruf pertamanya. Lalu huruf pertama tersebut akan dibandingkan satu persatu dengan huruf yang ada di dalam puzzle.
4. Ketika mentraversal satu persatu huruf dalam puzzle, apabila bertemu dengan huruf yang sama dengan huruf pertama kata yang dicari maka akan dicek ke delapan arah, mulai dari ATAS, ATAS KANAN, KANAN, KANAN BAWAH, BAWAH, KIRI BAWAH, KIRI, dan terakhir KIRI ATAS.
5. Pada setiap arah akan dibandingkan huruf kedua, ketiga hingga ke-n dari kata yang dicari.
6. Apabila tidak *match* pada suatu arah, maka akan dicari pada arah lain, hingga delapan arah semua dicek. Apabila tidak ketemu, akan lanjut mencari ke huruf-huruf selanjutnya pada puzzle.
7. Apabila sampai akhir kata tidak ditemukan, maka pencarian kata tersebut gagal.
8. Apabila sampai ke-n (Panjang kata yang dicari) dan masih sama, maka kata ditemukan dan pencarian kata tersebut berhenti.
9. Ulang semua proses kembali ke langkah (3) hingga semua kata selesai dicari.

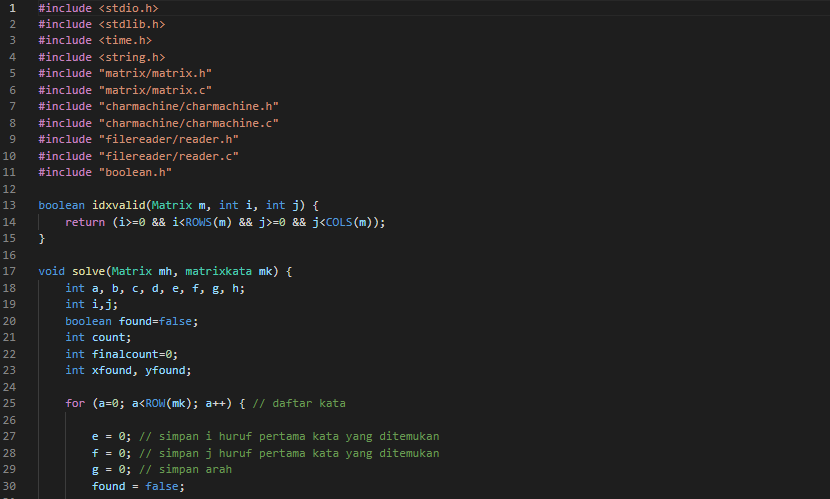
# BAB II *SOURCE PROGRAM* DALAM BAHASA C

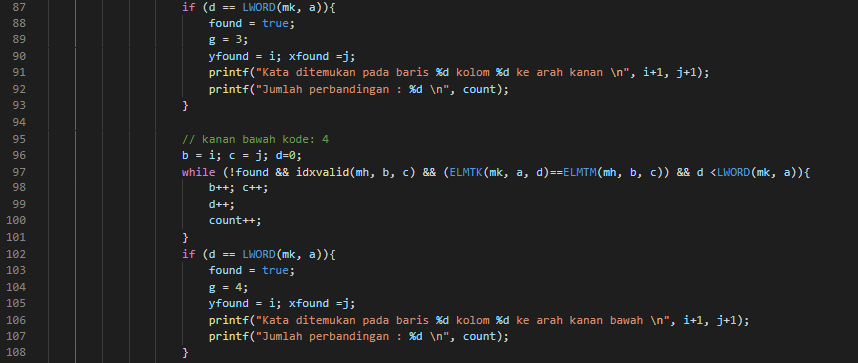
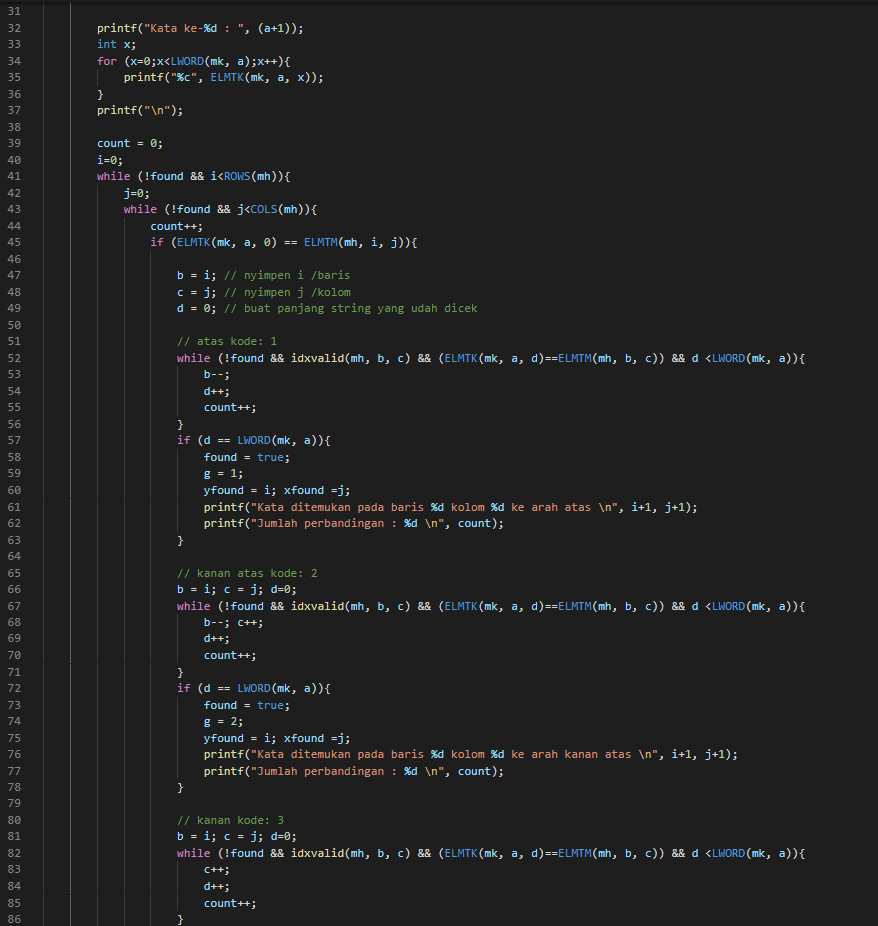
Pada pengerjaannya, program ditulis dalam bahasa C. Program secara lengkap dapat diakses pada pranala . . . Pada bab ini akan ditampilkan beberapa bagian potongan program guna memenuhi spesifikasi dari Tugas Kecil 1 IF2211 Strategi Algoritma.

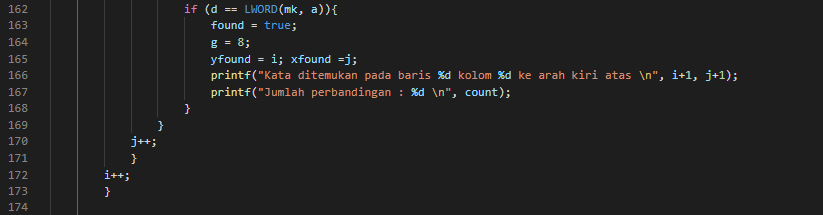
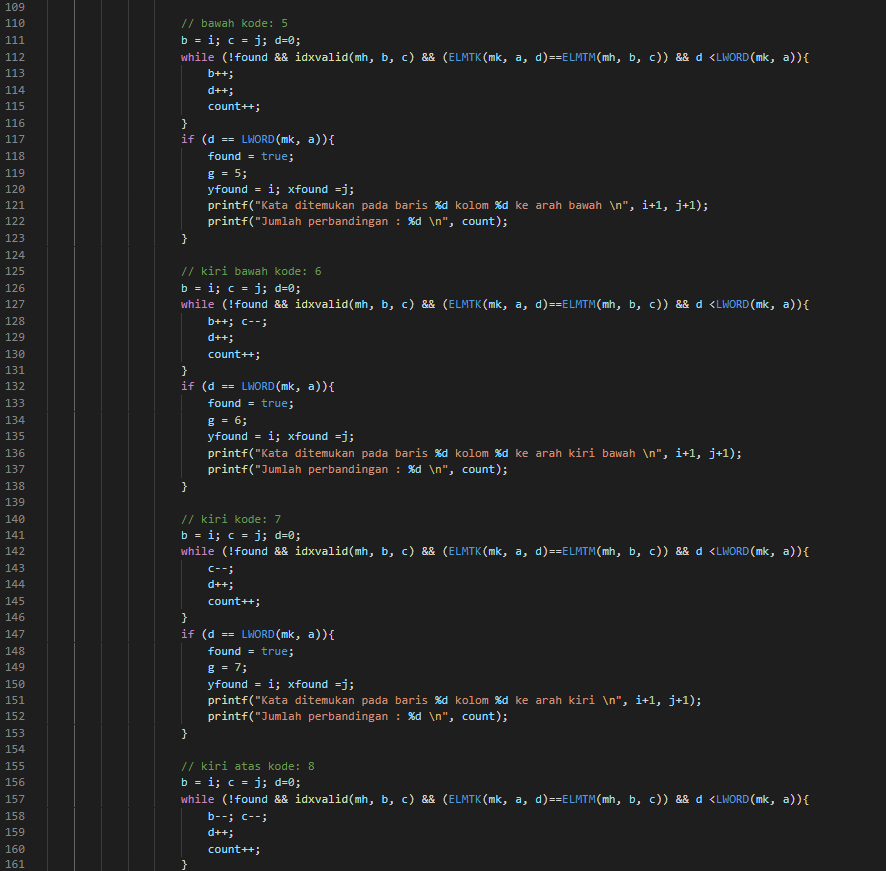
## Modularity Program

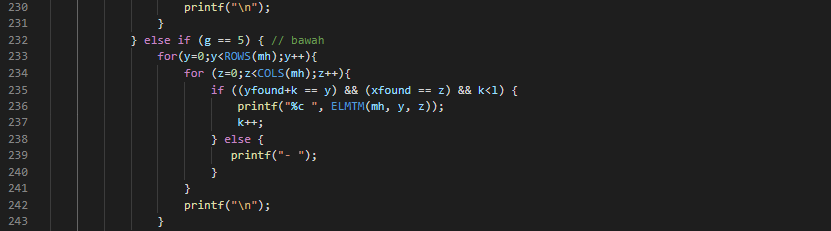
1. ├───src
2. │   ├───charmachine
3. │   │   ├───charmachine.c
4. │   │   └───charmachine.h
5. │   ├───filereader
6. │   │   └───reader.c
7. │   │   └───reader.h
8. │   ├───matrix
9. │   │   ├───matrix.c
10. │   │   └───matrix.h
11. │   ├───boolean.h
12. │   └───main.c
13. └───test
14. └───testcase.txt

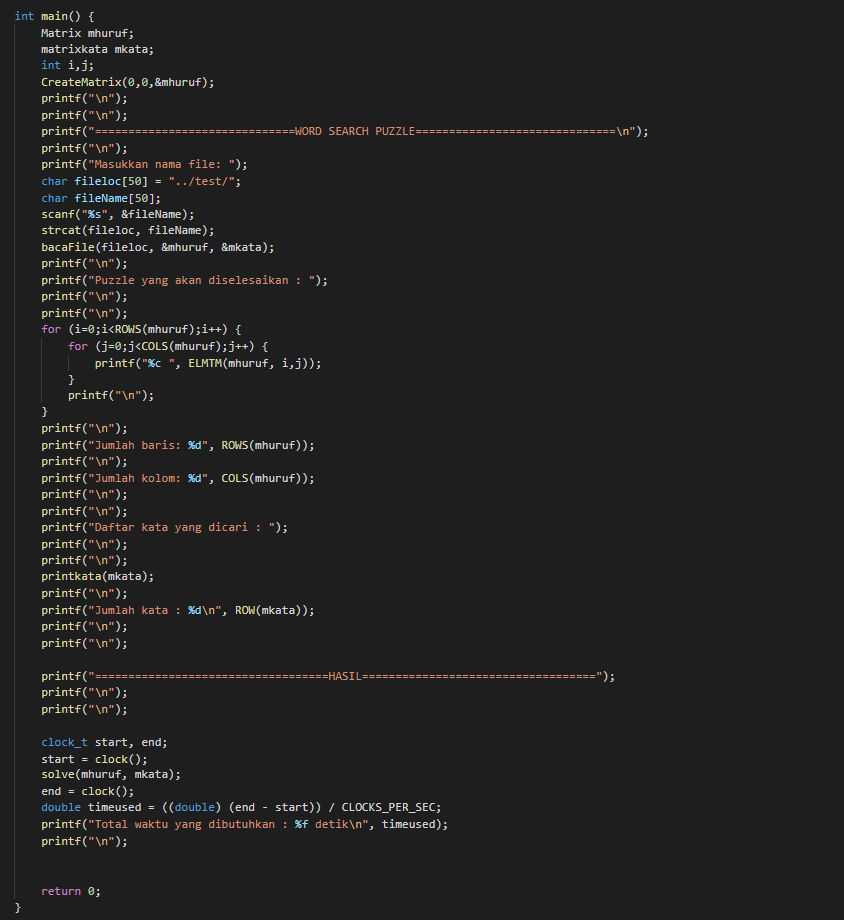
## main.c



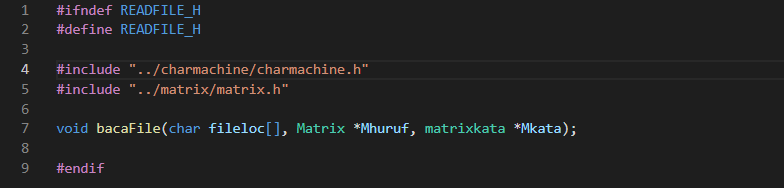




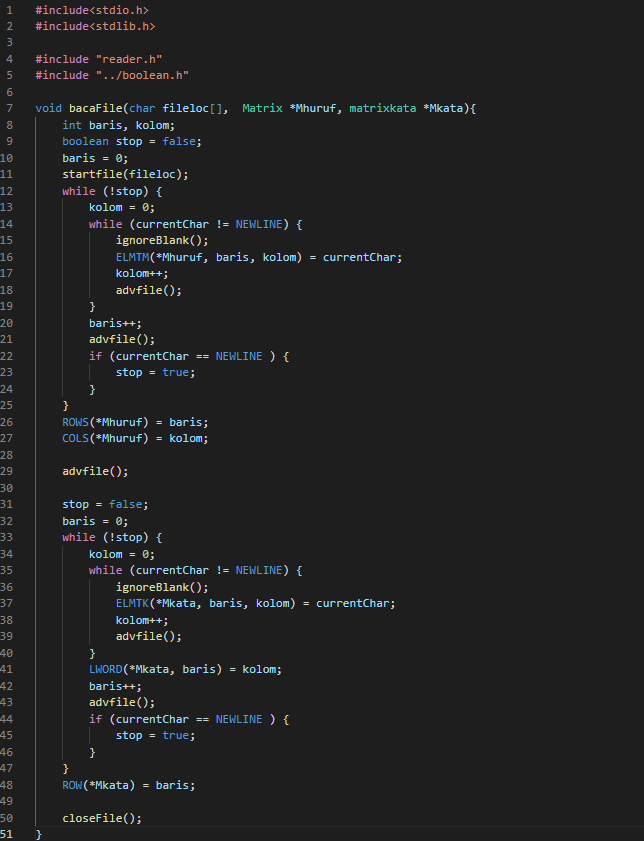


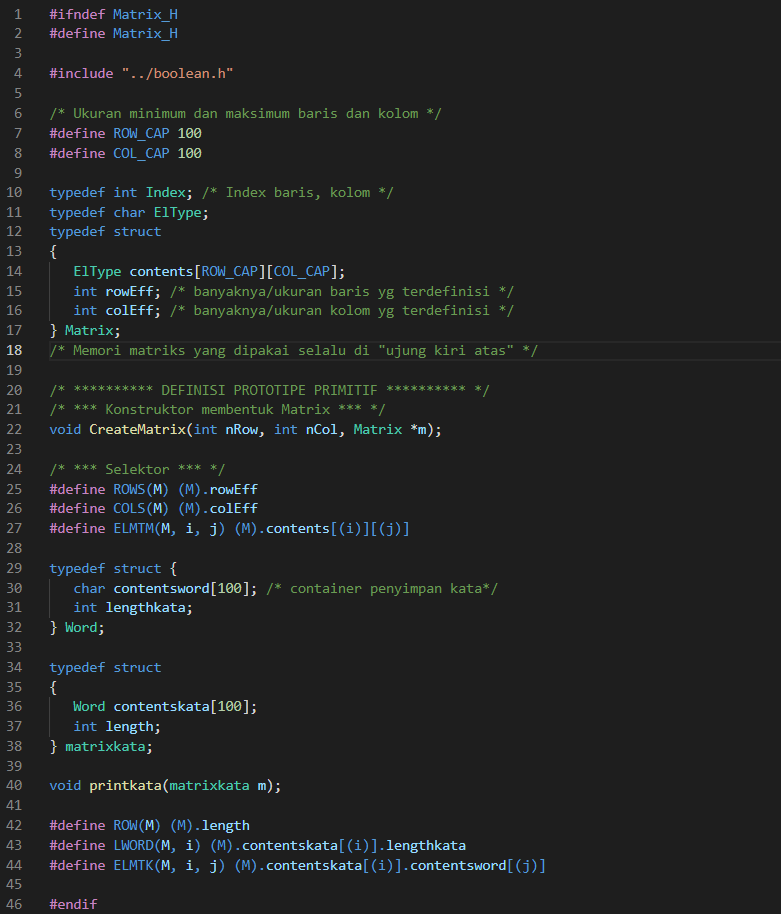
## reader.h



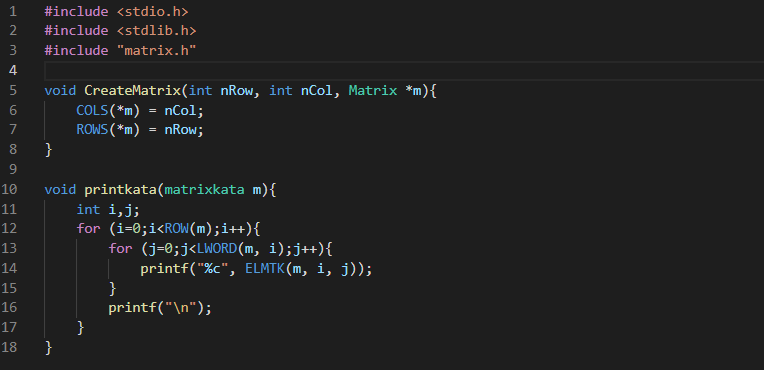
## reader.c



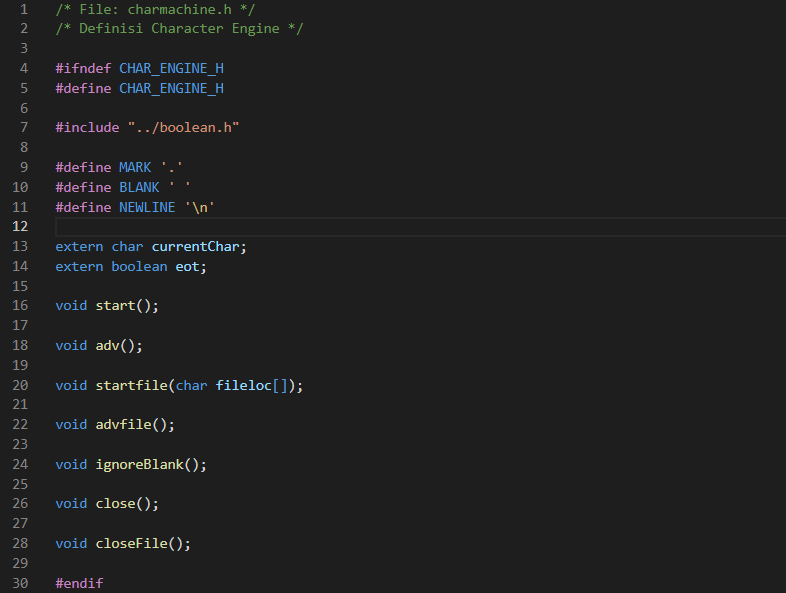
## matrix.h



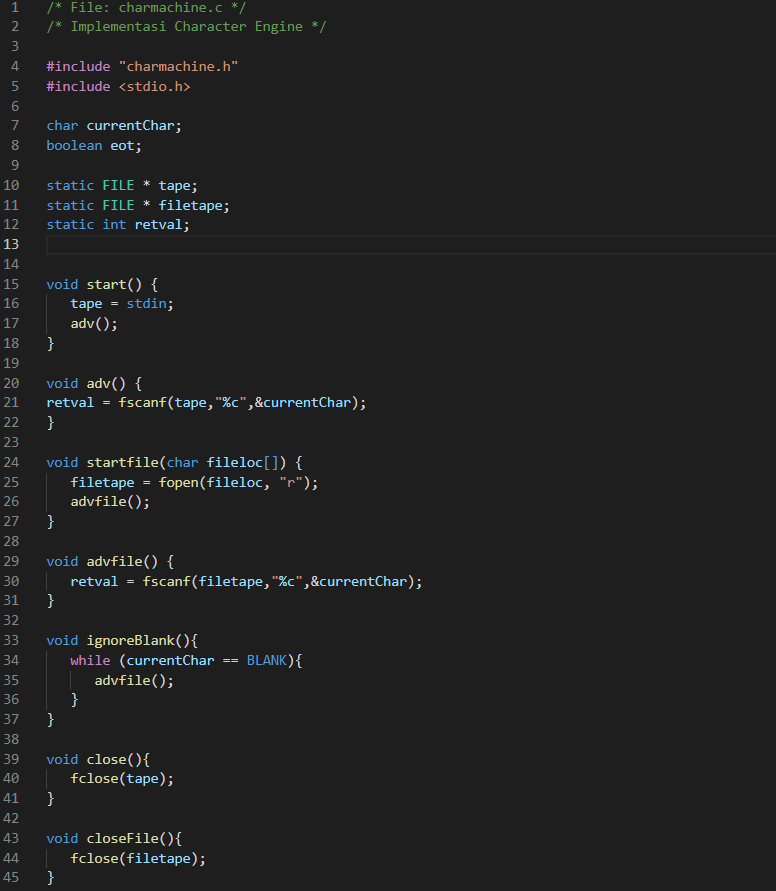
## matrix.c



## charmachine.h



## charmachine.c



# BAB III CONTOH *INPUT* DAN *OUTPUT* PROGRAM

## testcase1.txt

## testcase2.txt

## testcase3.txt

## testcase4.txt

## testcase5.txt

## testcase6.txt

## testcase7.txt

## testcase8.txt

# BAB IV KESIMPULAN DAN SARAN

## Kesimpulan

Program untuk menyelesaikan *Word Search Puzzle* yang telah penulis buat dapat berjalan dengan baik karena dapat memenuhi semua spesifikasi tugas yang diberikan. Namun mungkin saja masih ada beberapa kesalahan yang tidak saya sadari. Berikut adalah tabel yang diperlukan untuk mempermudah penilaian.

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi tanpa kesalahan (no syntax error). | ✓ |  |
| 1. Program berhasil *running.* | ✓ |  |
| 1. Program dapat membaca file masukan dan menuliskan luaran. | ✓ |  |
| 1. Program berhasil menemukan semua kata di dalam puzzle. | ✓ |  |

## Saran

Tentu saja program yang penulis buat masih sangat sederhana dan jauh dari kata sempurna. Masih banyak sekali hal yang bisa dilakukan untuk memperbaiki dan memperbagus program yang penulis buat, misalnya dengan menambahkan antarmuka GUI untuk mempermudah peengguna atau fitur-fitur lainnya.

# REFERENSI

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/stima21-22.htm>

<https://www.geeksforgeeks.org/how-to-measure-time-taken-by-a-program-in-c/>

<https://devdocs.io/c/>