**Laporan Tugas Kecil**

**IF 2211 Strategi Algoritma**

***Word Search Puzzle***



Nama : Angelica Winasta Sinisuka

NIM : 13520097

Kelas : K01

**Program Studi Teknik Informatika**

**Sekolah Teknik Elektro dan Informaika**

**Institut Teknologi Bandung**

**2022**

# **Langkah-langkah Pencarian Kata di Puzzle Menggunakan Algoritma Brute Force**

1. Sebelum melakukan pencarian kata, program menerima input berupa nama file dari user sesuai dengan directory lokasi file tersebut terhadap program. Diasumsikan isi dari file selalu benar sesuai spesifikasi.
2. Program akan memetakan puzzle ke dalam sebuah matrix berukuran puzzle dari file. Kemudian keyword akan dipetakan ke dalam sebuah array.
3. Langkah pencocokan kata dilakukan secara sekuensial. Mula-mula ditelusuri dari kanan ke kiri kemudian lanjut ke bawah pada matriks puzzle sampai:
   * karakter pertama kata kunci sama, akan mulai dibandingkan tiap char keyword dengan matriks puzzle dimulai arah ke kanan, bawah, kiri, atas, serong atas kiri, serong bawah kanan, serong kiri bawah, dan serong atas bawah.
   * dijumpai karakter yang tidak sama
4. Bila tidak ditemukan kata tersebut dalam puzzle, maka akan lanjut ke keyword berikutnya.
5. Jika sampai akhir kata keyword cocok dengan matrix, maka akan mengeluarkan output letak kata tersebut dalam puzzle.

# **Source Code**

Program ini menggunakan Bahasa c++.

/\*Nama: Angelica Winasta Sinisuka

NIM: 13520097

Tanggal: 24 Januari 2022

Deskripsi: Membuat program word search menggunakan algoritma brute force dengan bahasa c++\*/

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#include <fstream>

#include <string>

#include <cmath>

#include <chrono>

using namespace std;

int main(){

    string filename;

    cout << "Put in file name (ending with .txt): ";

    cin >> filename;

    ifstream theFile(filename);

    string lines;

    string puzzle[100];

string key[100];

    int countp = 0;

    int countk = 0;

    bool nextline = true;

    // Mengambil tiap line pada txt file

    while (getline(theFile, lines)){

        if (lines[1] != ' '){

            nextline = false;

        }

        if (nextline){

            puzzle[countp] = lines;

            countp++;

        }

        else{

            key[countk] = lines;

            cout << key[countk] << endl;

            countk++;

        }

    }

    // Memproses tiap line ke tipe data yang diassign, puzzle dan keyword

    // hasil: matrix

    // keyword: array[0..jmlkey]

    char n[puzzle[0].length()];

    int i = 0;

    int j = 0;

    int k = 0;

    int jumlah = puzzle[0].length();

    char hasil[countp-1][jumlah];

    while (i < countp-1){

        k = 0;

        for (j = 0; j < jumlah; j++){

            if (puzzle[i][j] != ' '){

                hasil[i][k] = puzzle[i][j];

                k++;

            }

        }

        i++;

    }

    // starting time

    auto mulai = std::chrono::high\_resolution\_clock::now();

    // menyocokan keyword dengan puzzle

    i = 0;

    bool samefirstletter = false;

    while (i < countk){ // traversal untuk mengecek tiap keyword secara sekuensial

        bool continues = true;

        int totalbanding = 0;

        int p = 0;

        while (p < countp-1 && continues){

            int q = 0;

            while ((q<k) && continues){

                totalbanding++;

                if (key[i][0] == hasil[p][q]){

                    samefirstletter = true;

                }

                else{

                    samefirstletter = false;

                }

                if (samefirstletter){

                    int valid = 0;

                    bool valid1 = true;

                    bool valid2 = false;

                    bool valid3 = false;

                    bool valid4 = false;

                    bool valid5 = false;

                    bool valid6 = false;

                    bool valid7 = false;

                    bool valid8 = false;

                    bool validtes = true;

                    int count = 1;

                    j = 1;

                    while (j < key[i].length()&& valid == 0 && validtes ){

                        totalbanding++;

                        if (valid1){// right

                            valid1 = key[i][j] == hasil[p][q+j];

                            if (valid1 == false || q+j > k){

                                valid2 = true;

                                valid = 0;

                                count = 1;

                                j = 1;

                            }

                            else{

                                count++;

                                if (count == key[i].length()){

                                    valid = 1;

                                }

                            }

                        }

                        if (valid2){// down

                            valid2 = key[i][j] == hasil[p+j][q];

                            if (valid2 == false || p+j > countp-1){

                                valid3 = true;

                                valid = 0;

                                count = 1;

                                j = 1;

                            }

                            else{

                                count++;

                                 if (count == key[i].length()){

                                    valid = 2;

                                }

                            }

                        }

                        if (valid3){// left

                            valid3 = key[i][j] == hasil[p][q-j];

                            if (valid3 == false || q-j < 0){

                                valid4 = true;

                                valid = 0;

                                count = 1;

                                j = 1;

                            }

                            else{

                                count++;

                                if (count == key[i].length()){

                                    valid = 3;

                                }

                            }

                        }

                        if (valid4){ // up

                            valid4 = key[i][j] == hasil[p-j][q];

                            if (valid4 == false || p-j < 0){

                                valid5 = true;

                                count = 1;

                                valid = 0;

                                j = 1;

                            }

                            else{

                                count++;

                                if (count == key[i].length()){

                                    valid = 4;

                                }

                            }

                        }

                        if (valid5){// up left

                            valid5 = key[i][j] == hasil[p-j][q-j];

                            if (valid5 == false || p-j < 0 || q-j < 0){

                                valid6 = true;

                                count = 1;

                                valid = 0;

                                j = 1;

                            }

                            else{

                                count++;

                                if (count == key[i].length()){

                                    valid = 5;

                                }

                            }

                        }

                        if (valid6){// down right

                            valid6 = key[i][j] == hasil[p+j][q+j];

                            if (valid6 == false || p+j > countp-1 || q+j > k ){

                                valid7 = true;

                                count = 1;

                                valid = 0;

                                j = 1;

                            }

                            else{

                                count++;

                                if (count == key[i].length()){

                                    valid = 6;

                                }

                            }

                        }

                        if (valid7){// down left

                            valid7 = key[i][j] == hasil[p+j][q-j];

                            if (valid7 == false  || p+j > countp-1 || q-j < 0){

                                valid8 = true;

                                count = 1;

                                valid = 0;

                                j = 1;

                            }

                            else{

                                count++;

                                if (count == key[i].length()){

                                    valid = 7;

                                }

                            }

                        }

                        if (valid8){// up right

                            valid8 = key[i][j] == hasil[p-j][q+j];

                            if (valid8 == false || p-j < 0 || q+j > k){

                                validtes = false;

                            }

                            else{

                                count++;

                                if (count == key[i].length()){

                                    valid = 8;

                                }

                            }

                        }

                        j++;

                    }

                // printing answer

                    if (valid <= 8 && valid >= 1){

                        continues = false;

                        char jawaban [countp-1][k];

                        for (int m = 0; m < countp-1; m++){

                            for (int n = 0; n < k; n++){

                                jawaban [m][n] = '-';

                            }

                        }

                        for (int l = 0; l < key[i].length();l++){

                            if (valid == 1){

                                jawaban[p][q+l] = key[i][l];

                            }

                            else if (valid == 2){

                                jawaban[p+l][q] = key[i][l];

                            }

                            else if (valid == 3){

                                jawaban[p][q-l] = key[i][l];

                            }

                            else if (valid == 4){

                                jawaban[p-l][q] = key[i][l];

                            }

                            else if (valid == 5){

                                jawaban[p-l][q-l] = key[i][l];

                            }

                            else if (valid == 6){

                                jawaban[p+l][q+l] = key[i][l];

                            }

                            else if (valid == 7){

                                jawaban[p+l][q-l] = key[i][l];

                            }

                            else if (valid == 8){

                                jawaban[p-l][q+l] = key[i][l];

                            }

                        }

                        for (int m = 0; m < countp-1; m++){

                            for (int n = 0; n < k; n++){

                                printf("%c ", jawaban[m][n]);

                            }

                            printf("\n");

                        }

                        cout << "Total Perbandingan Kata \"" << key[i] << "\": " << totalbanding << endl;

                    }

                }

                q++;

            }

            p++;

        }

        i++;

        samefirstletter = false;

    }

    auto akhir = std::chrono::high\_resolution\_clock::now();

    auto irisan = std::chrono::duration\_cast<std::chrono::nanoseconds>(akhir - mulai);

    printf("\nTotal Waktu yang Dibutuhkan: %.4f detik\n", irisan.count()\*1e-9);

    if (countk == '\0'){

        printf("Total Kata yang Dicari: 0\n\n");

        printf("Make sure file name is correct!\n");

    }

    else{

        printf("Total Kata yang Dicari: %d\n", countk+1);

    }

}

# **Test Case**

|  |  |  |
| --- | --- | --- |
| NO | Test Case | Output Program |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4. |  |  |
| 5 |  |  |
| 6 |  |  |

|  |  |  |
| --- | --- | --- |
| 7 |  | A picture containing background pattern  Description automatically generated |
| 8 |  |  |

Tabel 1. 3 Test case word puzzle

# Cek List Program

3.1 Check List

Table

Description automatically generated

