LAPORAN TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA

SEMESTER GENAP – 2021/2022

13520111 – RIZKY AKBAR ASMARAN

1. PENJELASAN ALGORITMA BRUTE FORCE
2. SOURCE CODE PROGRAM

|  |  |
| --- | --- |
| Procedure untuk baca masukan puzzle | Procedure untuk print puzzle |
| void bacaPuzzle(string filename){      ifstream puzzleReader(filename);      string baris;      if(puzzleReader.is\_open()){          while(getline(puzzleReader,baris)){              if(N == 0){                  M = (baris.size()+1)/2;              }              N++;              vector<char>row;              for(int i = 0; i < baris.size(); i+=2){                  row.push\_back(baris[i]);              }              puzzle.push\_back(row);          }          puzzleReader.close();      }  } | void printPuzzle(){      cout << "Baris: " << N << "; Kolom: " << M <<endl;      for (int i = 0; i < N; i++){          vector<char> row = puzzle[i];          for(int j = 0; j < M; j++){              cout << row[j] << " ";          }          cout << endl;      }  } |
| Function untuk validasi masukan dari puzzle hanya dari huruf saja | Function minimum |
| bool tervalidasi(string str){      int count = 0;      for(int i = 0; i < str.size(); i++){          if(str[i] < 'A' || str[i] > 'Z'){              return false;          }else{              count++;          }      }      return count >= 0 && count <= 50;  } | int minimum(int x, int y){      if(x < y){          return x;      }      return y;  } |
| Procedure untuk membaca list jawaban puzzle | Procedure untuk print list jawaban puzzle |
| void bacaListJawaban(string filename){      ifstream wordlistReader(filename);      string kata;      if(wordlistReader.is\_open()){          while(getline(wordlistReader, kata)){              if(tervalidasi(kata)){                  listKata.insert(kata);              }          }          wordlistReader.close();      }  } | void printListJawaban(){      cout << "List Jawaban: \n";      for(auto const& i: listKata){          cout << i << " \n";      }      cout << endl;  } |
| Procedure untuk mengecek huruf pada puzzle secara horizontal [-] | Procedure untuk mengecek huruf pada puzzle secara vertical [|] |
| void horizontal(int x, int y){      string tester1 = "";      string tester2 = "";      for(int i = 0; i < minimum(M-y, 50); i++){          tester1 = tester1 + puzzle[x][y+i];          tester2 = puzzle[x][y+i] + tester2;          if((i+1) >= 0){              checker\_and\_print(tester1, x, y, 2);              checker\_and\_print(tester2, x, y, 6);          }      }  } | void vertikal(int x, int y){      string tester1 = "";      string tester2 = "";      for(int i = 0; i < minimum(N-x, 50); i++){          tester1 = tester1 + puzzle[x+i][y];          tester2 = puzzle[x+i][y] + tester2;          if((i+1) >= 0){              checker\_and\_print(tester1, x, y, 4);              checker\_and\_print(tester2, x, y, 0);          }      }  } |
| Procedure untuk mengecek huruf pada puzzle secara diagonal [\] | Procedure untuk mengecek huruf pada puzzle secara diagonal [/] |
| void diagonal1(int x, int y){      string tester1 = "";      string tester2 = "";      for(int i = 0; i < minimum(50, minimum(N-x, M-y)); i++){          tester1 = tester1 + puzzle[x+i][y+i];          tester2 = puzzle[x+i][y+i] + tester2;          if((i+1) >= 0){              checker\_and\_print(tester1, x, y, 3);              checker\_and\_print(tester2, x, y, 7);          }      }  } | void diagonal2(int x, int y){      string tester1 = "";      string tester2 = "";      for(int i = 0; i < minimum(50, minimum(N-x, y+1)); i++){          tester1 = tester1 + puzzle[x+i][y-i];          tester2 = puzzle[x+i][y-i] + tester2;          if((i+1) >= 0){              checker\_and\_print(tester1, x, y, 5);              checker\_and\_print(tester2, x, y, 1);          }      }  } |
| Procedure untuk mengecek setiap kata pada puzzle dengan list dari jawaban dan akan membuat output sesuai dengan ditemukannya kata tersebut. | |
| void checker\_and\_print(string x, int x\_idx, int y\_idx, int dir){      char jawaban[N][M];      int i, j, counter, x2, y2;      perbandingan += 1;      if(listKata.count(x)){          for (i = 0; i < N; i++){              for (j = 0; j < M; j++) {                  jawaban[i][j] = '-';              }          }          if (dir == 0) {              x2 = x\_idx + x.length()-1;              y2 = y\_idx;              counter = 0;              while (counter < x.length()) {                  jawaban[x2][y2] = x[counter];                  counter += 1;                  x2 -= 1;              }          }          else if (dir == 1) {              x2 = x\_idx + x.length()-1;              y2 = y\_idx - x.length()+1;              counter = 0;              while (counter < x.length()) {                  jawaban[x2][y2] = x[counter];                  counter += 1;                  x2 -= 1;                  y2 += 1;              }          }          else if (dir == 2) {              x2 = x\_idx;              y2 = y\_idx;              counter = 0;              while (counter < x.length()) {                  jawaban[x2][y2] = x[counter];                  counter += 1;                  y2 += 1;              }          }          else if (dir == 3) {              x2 = x\_idx ;              y2 = y\_idx ;              counter = 0;              while (counter < x.length()) {                  jawaban[x2][y2] = x[counter];                  counter += 1;                  y2 += 1;                  x2 += 1;              }          }          else if (dir == 4) {              x2 = x\_idx;              y2 = y\_idx;              counter = 0;              while (counter < x.length()) {                  jawaban[x2][y2] = x[counter];                  counter += 1;                  x2 += 1;              }          }          else if (dir == 5) {              x2 = x\_idx;              y2 = y\_idx;              counter = 0;              while (counter < x.length()) {                  jawaban[x2][y2] = x[counter];                  counter += 1;                  x2 += 1;                  y2 -= 1;              }          }          else if (dir == 6) {              x2 = x\_idx;              y2 = y\_idx + x.length()-1;              counter = 0;              while (counter < x.length()) {                  jawaban[x2][y2] = x[counter];                  counter += 1;                  y2 -= 1;              }          }          else if (dir == 7) {              x2 = x\_idx + x.length()-1;              y2 = y\_idx + x.length()-1;              counter = 0;              while (counter < x.length()) {                  jawaban[x2][y2] = x[counter];                  counter += 1;                  x2 -= 1;                  y2 -= 1;              }          }          cout << "================================\n";          for (i = 0; i < N; i++) {              for (j = 0; j < M; j++) {                  cout << " " <<jawaban[i][j] << " " ;              }              cout << endl;          }          kata2.insert(x);      }  } | |
| Procedure untuk mengecek puzzle dari setiap arah | Program Utama |
| void solver(){      for (int x = 0; x < N; x++){          for(int y = 0; y < M; y++){              horizontal(x,y);              vertikal(x,y);              diagonal1(x,y);              diagonal2(x,y);          }      }  } | int main(){      bacaListJawaban("test1.txt");      bacaPuzzle("test.txt");      printPuzzle();      uint64\_t awal = duration\_cast<milliseconds>(system\_clock::now().time\_since\_epoch()).count();      solver();      uint64\_t akhir = duration\_cast<milliseconds>(system\_clock::now().time\_since\_epoch()).count();      cout << endl << "Proses selesai dalam waktu: "<< akhir-awal << " ms\n" << endl;      cout << "Total Perbandingan Kata : " << perbandingan << endl;      cout << "\n";      printListJawaban();      return 0;  } |

1. SCREENSHOT DARI INPUT DAN OUTPUT
2. LINK SOURCE CODE
3. CHECKLIST

|  |  |  |
| --- | --- | --- |
| Poin | YA | TIDAK |
| 1. Program berhasil dikompilasi tanpa kesalahan (no syntax error) | √ |  |
| 1. Program berhasil *running* | √ |  |
| 1. Program dapat membaca berkas input dan menuliskan luaran | √ |  |
| 1. Program berhasil menemukan semua kata di dalam puzzle | √ |  |