Laporan Tugas Kecil I Strategi Algoritma

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi tanpa kesalahan (no syntax error) |  |  |
| 2. Program berhasil running |  |  |
| 3. Program dapat membaca file masukan dan menuliskan luaran. |  |  |
| 4. Program berhasil menemukan semua kata di dalam puzzle |  |  |

Langkah-langkah

Algoritma pembacaan file:

- Pada pembacaan file ini, terdapat 2 state / mode program, yang pertama mode untuk membaca word puzzle dan yang kedua ialah mode membaca kata-kata yang akan dicari.

- Data word puzzle disimpan pada vektor 2 dimensi berisi pasangan karakter untuk karakter word puzzle dan integer untuk kode warna huruf ketika ditampilkan. Sedangkan data kata-kata yang akan dicari disimpan pada vektor string.

- Buka file dengan alamat berdasarkan masukan pengguna, jika file tidak ditemukan, akan ditampilkan pesan “File not found” dan program berhenti.

- Lakukan iterasi setiap baris pada file, jika baris tersebut panjangnya kurang dari atau sama dengan 1, maka ganti ke state / mode 2

- Jika baris panjangnya lebih dari 1 dan sedang di state / mode 1, maka kita lakukan pembacaan karakter yang bukan whitespace dan simpan pada vektor pasangan karakter dan integer. Setelah selesai pembacaan satu baris, maka simpan pada vektor 2 dimensi sebelumnya.

- Jika baris panjangnya lebih dari 1 dan sedang di state / mode 2, maka kita langsung masukan string baris pembacaan ke vektor string sebelumnya.

Algoritma pemrosesan:

- Iterasi setiap kata yang akan dicari pada word puzzle

- Untuk setiap iterasi, kita cari kata tersebut pada word puzzle secara delapan arah secara berurutan. Delapan arah tersebut ialah horizontal ke kanan, horizontal ke kiri, vertical ke bawah, vertical ke atas, diagonal utama ke kanan bawah, diagonal utama ke kiri atas, diagonal samping ke kiri bawah dan diagonal samping ke kanan atas.

- Pencarian tersebut dengan cara mencari kata yang dicari saat itu pada setiap baris (jika horizontal), setiap kolom (jika vertical), atau setiap diagonal (jika diagonal) word puzzle.

- Pencarian pada setiap baris / kolom / diagonal tersebut diawali dengan menghitung berapa banyak kata dengan panjang sama dengan kata yang dicari, yang muat pada baris / kolom / diagonal tersebut. Untuk setiap nilai tersebut, maka dilakukan komparasi string dengan membandingkan karakter satu per satu.

- Jika sudah berbeda karakter, maka hentikan komparasi dan lanjut ke iterasi berikutnya

- Jika benar, maka ganti kode warna huruf-huruf pada word puzzle yang bersesuaian dengan kata yang dicari dan hentikan pencarian kata tersebut. Kode warna didapatkan dari indeks / urutan kata yang dicari.

- Jika sudah ditemukan di salah satu arah (horizontal / vertical / diagonal), maka tidak dilakukan pencarian kata yang sama pada arah yang berbeda.

- Jika sudah semua kata yang dicari ditemukan, maka akan ditampilkan ke layar huruf-huruf word puzzle dengan warna yang bersesuaian dengan kata-kata yang dicari, jika ada yang bersesuaian.

Algoritma output hasil

- Iterasi setiap baris dari data word puzzle (berupa vektor 2 dimensi dari pasangan karakter dan integer)

- Iterasi setiap kolom pada baris, dan tampilkan huruf ke layar berdasarkan kode warna huruf tersebut.

- Program ini hanya memiliki 13 kombinasi warna, 12 untuk hasil kata yang dicari dan 1 untuk huruf-huruf yang tidak memiliki kata yang bersesuaian. Oleh karena itu, jika terdapat lebih dari 12 kata yang dicari, maka akan terdapat warna yang sama pada hasil pencarian karena dilakukan modulo berdasarkan kode warna dan jumlah warna yang tersedia (12).

Source program

Program ini dibagi menjadi 4 bagian, yaitu: main program, file reader program, word puzzle solver program, serta utils program. Dengan semua program kecuali main program, memiliki header file-nya masing-masing. Pada program ini, diterapkan konsep pemrograman berbasis objek untuk mengisolasi method dan atribut, terutama atribut untuk data global objek tersebut yang tidak seharusnya diakses oleh program lain sehingga membuat pengembangan program menjadi lebih aman.

Main Program

#include "file\_reader/file\_reader.hpp"

#include "utils/utils.hpp"

#include "word\_puzzle\_solver/word\_puzzle\_solver.hpp"

#include <array>

#include <chrono>

#include <ctime>

#include <iostream>

#include <string>

#include <utility>

#include <vector>

int main(int argc, char \*\*argv) {

  std::vector<std::vector<std::pair<char, int>>> word\_puzzle;

  std::vector<std::string> words\_searched;

  if (argc < 2) {

    std::cout << "Please add arguments for file name" << std::endl;

    return 0;

  }

  FileReader fileReader = FileReader();

  if (!fileReader.readFile(std::string("./tests/") + argv[1], word\_puzzle,

                           words\_searched)) {

    return 0;

  }

  std::chrono::time\_point<std::chrono::system\_clock> start, end;

  start = std::chrono::system\_clock::now();

  WordPuzzleSolver wps = WordPuzzleSolver();

  wps.solve(word\_puzzle, words\_searched);

  end = std::chrono::system\_clock::now();

  Utils utils = Utils();

  utils.showOutput(word\_puzzle);

  std::cout << "COMPARISONS: " << wps.getComparisons() << std::endl;

  std::chrono::nanoseconds elapsed\_seconds = end - start;

  std::time\_t end\_time = std::chrono::system\_clock::to\_time\_t(end);

  std::cout << "elapsed time: " << elapsed\_seconds.count() << " nanoseconds\n";

  return 0;

}

File reader program

header file

#pragma once

#include <fstream>

#include <string>

#include <vector>

class FileReader {

private:

  std::ifstream fin;

public:

  bool readFile(std::string fileName,

                std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                std::vector<std::string> &words\_searched);

};

C++ file

#include "file\_reader.hpp"

#include <fstream>

#include <iostream>

#include <string>

#include <vector>

bool FileReader::readFile(

    std::string fileName,

    std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

    std::vector<std::string> &words\_searched) {

  bool isMode1 = true;

  std::string line;

  fin.open(fileName);

  if (!fin.is\_open()) {

    std::cout << "File not found" << std::endl;

    return false;

  }

  while (fin) {

    getline(fin, line);

    if (line.length() > 1) {

      if (isMode1) {

        std::vector<std::pair<char, int>> line\_vector;

        for (int i = 0; i < line.length(); i++) {

          if (line[i] != ' ') {

            line\_vector.push\_back({line[i], -1});

          }

        }

        word\_puzzle.push\_back(line\_vector);

      } else {

        words\_searched.push\_back(line);

      }

    } else {

      if (isMode1) {

        isMode1 = false;

      } else {

        break;

      }

    }

  }

  fin.close();

  return true;

}

Word Puzzle Solver program

header file

#pragma once

#include <iostream>

#include <string>

#include <vector>

class WordPuzzleSolver {

private:

  int idxWord;

  long long comparisons = 0;

  bool solvePerWord(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                    std::string &word\_searched);

  int inline calcIndexWord(bool isReversed, int idx, int word\_length) {

    return isReversed ? word\_length - idx - 2 : idx;

  }

  bool

  solveHorizontal(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                  std::string &word\_searched, bool isReversed);

  bool

  solveVertical(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                std::string &word\_searched, bool isReversed);

  bool

  solveMainDiagonal(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                    std::string &word\_searched);

  bool solveCrossDiagonal(

      std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

      std::string &word\_searched);

  bool

  checkMainDiagonal(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                    std::string &word\_searched, int rowStart, int colStart,

                    bool isReversed);

  bool checkCrossDiagonal(

      std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

      std::string &word\_searched, int rowStart, int colStart, bool isReversed);

  void addColor(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                const int row, const int col, const int len, const char mode);

public:

  void solve(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

             std::vector<std::string> &words\_searched);

  long long inline getComparisons() { return comparisons; }

};

C++ file

#pragma once

#include <iostream>

#include <string>

#include <vector>

class WordPuzzleSolver {

private:

  int idxWord;

  long long comparisons = 0;

  bool solvePerWord(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                    std::string &word\_searched);

  int inline calcIndexWord(bool isReversed, int idx, int word\_length) {

    return isReversed ? word\_length - idx - 2 : idx;

  }

  bool

  solveHorizontal(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                  std::string &word\_searched, bool isReversed);

  bool

  solveVertical(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                std::string &word\_searched, bool isReversed);

  bool

  solveMainDiagonal(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                    std::string &word\_searched);

  bool solveCrossDiagonal(

      std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

      std::string &word\_searched);

  bool

  checkMainDiagonal(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                    std::string &word\_searched, int rowStart, int colStart,

                    bool isReversed);

  bool checkCrossDiagonal(

      std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

      std::string &word\_searched, int rowStart, int colStart, bool isReversed);

  void addColor(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

                const int row, const int col, const int len, const char mode);

public:

  void solve(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle,

             std::vector<std::string> &words\_searched);

  long long inline getComparisons() { return comparisons; }

};

Utils program

header file

#pragma once

#include <iostream>

#include <string>

#include <vector>

#define RESET "\033[0m"

#define RED "\033[31m"                /\* Red \*/

#define GREEN "\033[32m"              /\* Green \*/

#define YELLOW "\033[33m"             /\* Yellow \*/

#define BLUE "\033[34m"               /\* Blue \*/

#define MAGENTA "\033[35m"            /\* Magenta \*/

#define CYAN "\033[36m"               /\* Cyan \*/

#define BOLDRED "\033[1m\033[31m"     /\* Bold Red \*/

#define BOLDGREEN "\033[1m\033[32m"   /\* Bold Green \*/

#define BOLDYELLOW "\033[1m\033[33m"  /\* Bold Yellow \*/

#define BOLDBLUE "\033[1m\033[34m"    /\* Bold Blue \*/

#define BOLDMAGENTA "\033[1m\033[35m" /\* Bold Magenta \*/

#define BOLDCYAN "\033[1m\033[36m"    /\* Bold Cyan \*/

class Utils {

private:

  const int codeAmount = 12;

  const std::string colorCode[12] = {

      RED,     GREEN,     YELLOW,     BLUE,     MAGENTA,     CYAN,

      BOLDRED, BOLDGREEN, BOLDYELLOW, BOLDBLUE, BOLDMAGENTA, BOLDCYAN,

  };

  std::string inline getColor(int code) {

    return code == -1 ? RESET : colorCode[code % codeAmount];

  }

public:

  void showOutput(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle);

  void

  showCodeColor(std::vector<std::vector<std::pair<char, int>>> &word\_puzzle);

};

C++ file

#include "utils.hpp"

void Utils::showOutput(

    std::vector<std::vector<std::pair<char, int>>> &word\_puzzle) {

  for (int i = 0; i < word\_puzzle.size(); i++) {

    for (int j = 0; j < word\_puzzle[i].size(); j++) {

      std::cout << getColor(word\_puzzle[i][j].second) << word\_puzzle[i][j].first

                << " ";

    }

    std::cout << std::endl;

  }

  std::cout << RESET;

}

void Utils::showCodeColor(

    std::vector<std::vector<std::pair<char, int>>> &word\_puzzle) {

  for (int i = 0; i < word\_puzzle.size(); i++) {

    for (int j = 0; j < word\_puzzle[i].size(); j++) {

      std::cout << word\_puzzle[i][j].second << " ";

    }

    std::cout << std::endl;

  }

}

Skrinshut input dan output

Kategori ukuran kecil

Small 1

A picture containing arrow

Description automatically generated A picture containing text

Description automatically generated

Small 2

Text

Description automatically generated Text

Description automatically generated

Small 3

A screenshot of a computer

Description automatically generated with medium confidence Text

Description automatically generated

Kategori Ukuran Sedang

Medium 1

Text

Description automatically generated with medium confidence A picture containing arrow

Description automatically generated

Medium 2

A picture containing text

Description automatically generated Arrow

Description automatically generated

Medium 3

A picture containing shape

Description automatically generated Arrow

Description automatically generated with medium confidence

Kategori ukuran besar

Large 1

Background pattern

Description automatically generated A screenshot of a computer screen

Description automatically generated with medium confidence

Background pattern

Description automatically generated

Large 2

Background pattern

Description automatically generated A picture containing table

Description automatically generated

Shape

Description automatically generated with medium confidence

Large 3

Background pattern

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence

