Міністерство освіти і науки України

Національний технічний університет «Дніпровська політехніка»

ЗВІТ
з лабораторної роботи №1
дисципліни «Розробка мікросервісних систем на мові Golang»

Виконав: ст. гр. 123-20ск-1

Гладкий Сергій Сергійович

Прийняв:
Реута О.В.

Дніпро
2022

Task 1

“CSV Sorter” is a CLI application that allows sorting of its input presented as CSV-text.

Technical details

Required features:

1. The application runs as a CLI application.
2. It reads STDIN line by line. The end of the input is an empty line.
3. Each line is a list of comma-separated values (CSV). Each value is considered as a piece of text. The number of values is the same in each line.
4. The application sorts all lines alphabetically by the first value in each line.
5. The application prints the result immediately, when the user ends to enter input text (presses <Enter> at a new line).

Optional features (not required but appreciated):

1. The application supports options:

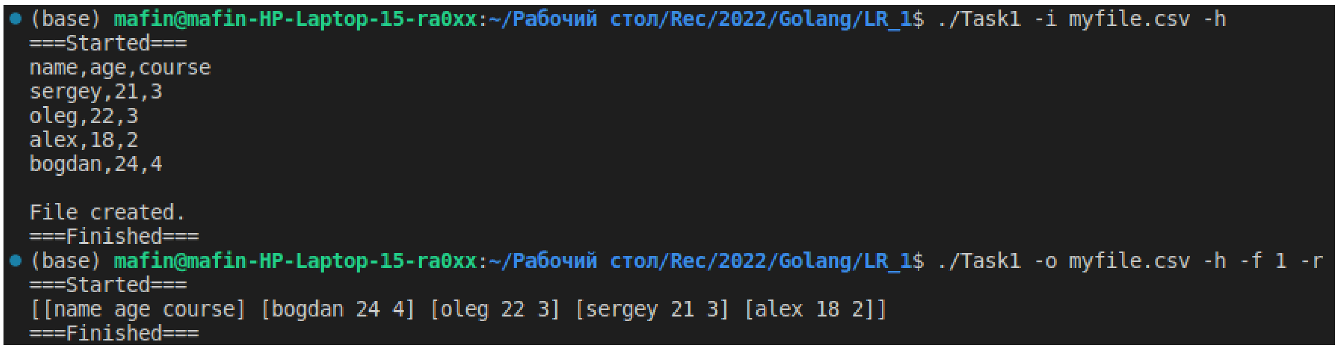
Option usage	**Meaning**
-i	file-name Use a file with the name **file-name** as an input.
-o	file-name Use a file with the name **file-name** as an output.
-h	The first line is a header that must be ignored during sorting but included in the output.
-f	**N** Sort input lines by value number **N**.
-r	Sort input lines in reverse order.

Program code:

```
package main

import (
	"bufio"
	"encoding/csv"
	"flag"
	"fmt"
	"os"
	"sort"
	"strings"
```

```go
)

func main() {
	var (
		inputFileName = flag.String("i", "", "Use a file with the
			name file-name as an input.")
		outputFileName = flag.String("o", "", "Use a file with the
			name file-name as an output.")
		ignoreHeader = flag.Bool("h", false, "The first line is a
			header that must be ignored during sorting but
			included in the output.")
		sortingField = flag.Int("f", 0, "Sort input lines by value
			number N.")
		reverseSort = flag.Bool("r", false, "Sort input lines in
			reverse order.")
	)

	flag.Parse()

	fmt.Println("===Started===")

	// Write to file
	if inputFileName != nil && *inputFileName != "" {
		records := writeRecords()
		sortCsvData(records, *ignoreHeader, *reverseSort,
		*sortingField)
		writeCsvFile(*inputFileName, records)
	}

	// Read from file
	if outputFileName != nil && *outputFileName != "" {
		records := readCsvFile(*outputFileName)
		sortCsvData(records, *ignoreHeader, *reverseSort,
		*sortingField)
		fmt.Println(records)
	}

	fmt.Println("===Finished===")
}

func readCsvFile(filePath string) [][]string {
	f, err := os.Open(filePath)
	if err != nil {
		fmt.Println(err)
		os.Exit(1)
	}
	defer f.Close()

	content := [][]string{}

	csvReader := csv.NewReader(f)
	records, err := csvReader.Read()
	for records != nil {
		if err != nil {
			fmt.Println("", err)
			os.Exit(1)
		}

		content = append(content, records)
		records, err = csvReader.Read()
	}

	return content
}

func sortCsvData(content [][]string, ignoreHeader, reverse bool,
field int) {
	if field > (len(content[0]) - 1) {
		fmt.Printf("Error: only %d column in this file.\n",
		len(content[0]))
		os.Exit(1)
	}
	if reverse {
		if ignoreHeader {
			sort.Slice(content[1:], func(i, j int) bool {
			return content[1:][i][field] > content[1:][j][field]
		})
		} else {
			sort.Slice(content, func(i, j int) bool {
			return content[i][field] > content[j][field]
			})
		}
	} else {
		if ignoreHeader {
			sort.Slice(content[1:], func(i, j int) bool {
			return content[1:][i][field] < content[1:][j][field]
			})
		} else {
				sort.Slice(content, func(i, j int) bool {
				return content[i][field] < content[j][field]
				})
			}
		}
	}

	func writeCsvFile(name string, data [][]string) {
		file, err := os.Create(name)
		if err != nil {
			fmt.Println("Unable to create file:", err)
			os.Exit(1)
		}
		defer file.Close()

		csvWriter := csv.NewWriter(file)
		err = csvWriter.WriteAll(data)

		if err != nil {
			fmt.Println("Unable to create file:", err)
			os.Exit(1)
		}

		fmt.Println("File created.")
	}

	func writeRecords() [][]string {
		s := bufio.NewScanner(os.Stdin)

		records := [][]string{}

		n := 0

		for s.Scan() {
			line := s.Text()
			if line == "" {
				break
			}
			row := strings.Split(line, ",")

			if n == 0 {
				n = len(row)
			}
```

```
            if n != len(row) {
                fmt.Printf("Error: row has %d column, but must
                    have %d\n", len(row), n)
                os.Exit(1)
            }

            records = append(records, row)
        }

        return records
    }
```



```
● (base) mafin@mafin-HP-Laptop-15-ra0xx:~/Рабочий стол/Rec/2022/Golang/LR_1$ ./Task1 -i myfile.csv -h
  ===Started===
  name,age,course
  sergey,21,3
  oleg,22,3
  alex,18,2
  bogdan,24,4

  File created.
  ===Finished===
● (base) mafin@mafin-HP-Laptop-15-ra0xx:~/Рабочий стол/Rec/2022/Golang/LR_1$ ./Task1 -o myfile.csv -h -f 1 -r
  ===Started===
  [[name age course] [bogdan 24 4] [oleg 22 3] [sergey 21 3] [alex 18 2]]
  ===Finished===
```

Picture 1 - Result