Додаток А

Міністерство Освіти і науки України
КПІ ім. Ігоря Сікорського
Кафедра ІПІ

ЗВІТ
з виконання лабораторної роботи № 1
з кредитного модуля
“Основи програмування-2. Методологія програмування”

Варіант № 9

Виконав:
студент 1-го курсу
гр. ІП-25 ФІОТ
Карпов Любомир Васильович

Київ 2023

Постановка задачі

Створити текстовий файл, рядки якого містять розділені пробілами слова, що складаються із цифр або символів. У кожному рядку, що містить числа, знайти найменше із них. Переписати такі рядки (що містять числа) у новий текстовий файл наступним чином: починається такий рядок знайденим найменшим числовим значенням, далі послідовно, через кому, записуються інші числові значення відповідного рядка у порядку, зворотному порядку їх слідування у рядку вхідного тексту, а потім усі інші слова цього рядка у порядку їхнього слідування. Вивести вміст вхідного і створеного файлів.

Текст програми

main.cpp

```
#include "FileStream.h"
#include "FilePointer.h"

std::string MODE = "-mode";
std::string FILE_POINTER_MODE = "FilePointer";
std::string FILE_STREAM_MODE = "FileStream";

std::string IN_FILE_NAME = "in.txt";
std::string OUT_FILE_NAME = "out.txt";

int main(int argc, char *argv[]) {

    std::cout << "Ctrl+B char: " << (char) 2 << '\n';

    if (argc >= 3 && argv[1] == MODE) { // Method choice
        if (argv[2] == FILE_POINTER_MODE) {
            file_pointer_main(IN_FILE_NAME.c_str(), OUT_FILE_NAME.c_str()); //
file pointer method
        } else if (argv[2] == FILE_STREAM_MODE) {
            file_stream_main(IN_FILE_NAME, OUT_FILE_NAME);  // file stream
method
        } else
            std::cout << "Incorrect mode!\n";
    } else {
        std::cout << "Missing arguments!\n";
    }

}
```

FileStream.h

```
#ifndef LAB1_FILESTREAM_H
#define LAB1_FILESTREAM_H
```

```
#include <fstream>
#include <iostream>

void file_stream_main(const std::string& in_file_name, const std::string&
out_file_name);

void print_from_file(const std::string& file_name);

void create_or_append_file(const std::string& file_name);

bool is_number(const std::string &s);

void task(const std::string& in_file_name, const std::string&
out_file_name);

#endif //LAB1_FILESTREAM_H
```

FileStream.cpp

```
#include "FileStream.h"


void file_stream_main(const std::string &in_file_name, const std::string
&out_file_name) {
    create_or_append_file(in_file_name); // Prev editing input file (or
creating if not exists)
    std::cout << "Input file:\n";
    print_from_file(in_file_name); // printing input file
    task(in_file_name, out_file_name); // processing task
    std::cout << "Output file:\n";
    print_from_file(out_file_name); // printing output file
}

void print_from_file(const std::string &file_name) {
    std::ifstream file(file_name);
    std::string line;

    while (getline(file, line)) {
        std::cout << line << '\n';
    }

    file.close();
}

void create_or_append_file(const std::string &file_name) {
    std::ifstream in_file(file_name);
    int mode = 0;
    if (in_file.is_open()) { // if file found print content and ask what to
do
        std::cout << "Found file, content:\n";
        in_file.close();
```

```
        print_from_file(file_name);

        std::cout << "To rewrite file enter 0, to mode 1, to not change 2:
";
        std::cin >> mode;
    }

    std::ofstream out_file;
    switch (mode) {
        case 0: {
            out_file.open(file_name); // rewrite mode
            break;
        }
        case 1: {
            out_file.open(file_name, std::ios::app); // append mode
            break;
        }
        case 2: {
            return; // not edit mode
        }
        default: {
            throw;
        }
    }

    std::cout << "Writing to file.\nIf you want to end input, go to next
line, press Ctrl+B and then Enter.\n";

    std::string line;
    while (true) {
        std::getline(std::cin, line);
        if (line[0] == 2) // 2 is code of Ctrl+B
            break;
        if (!line.empty())
            out_file << line << '\n';
    }

    out_file.close();

}

bool is_number(const std::string &s) {
    if (s.empty())
        return false;

    for (char i : s) {
        if (!isdigit(i)) // if one of chars is not digit it is word
            return false;
    }

    return true;
}
```

```
void task(const std::string &in_file_name, const std::string
&out_file_name) {
    std::ifstream in_file(in_file_name);
    std::ofstream out_file(out_file_name);
    std::string line;

    while (getline(in_file, line)) {
        bool min_found = false;
        int min;

        std::string str = line;
        while (true) { // finding min number
            int cur = str.find_first_of(' ');
            if (cur == -1) {
                if (is_number(str)) {
                    int num = std::stoi(str);
                    if (min_found) {
                        if (min > num)
                            min = num;
                    } else {
                        min = num;
                        min_found = true;
                    }
                }
                break;
            } else {

                std::string word = str.substr(0, cur);
                str.erase(0, cur + 1);

                if (is_number(word)) {
                    int num = std::stoi(word);
                    if (min_found) {
                        if (min > num)
                            min = num;
                    } else {
                        min = num;
                        min_found = true;
                    }
                }

            }
        }

        if (min_found) { // if min found write line to output file
            out_file << min << ' ';
            str = line;

            while (true) {
                int cur = str.find_last_of(' ');
                if (cur == -1) {

                    break;
```

```
                } else {
                    std::string word = str.substr(cur + 1, str.size());
                    str.erase(cur, str.size());

                    if (is_number(word)) {
                        int num = std::stoi(word);
                        if (min != num) {
                            out_file << num << ' ';
                        }
                    }
                }
            }
            str = line;
            while (true) {
                int cur = str.find_first_of(' ');
                if (cur == -1) {
                    if (!is_number(str)) {
                        out_file << str << ' ';
                    }
                    break;
                } else {
                    std::string word = str.substr(0, cur);
                    str.erase(0, cur + 1);
                    if (!is_number(word)) {
                        out_file << word << ' ';
                    }
                }
            }

            out_file << '\n';
        }

    }

    in_file.close();
    out_file.close();

}
```

FilePointer.h

```
#ifndef LAB1_FILEPOINTER_H
#define LAB1_FILEPOINTER_H

#include <iostream>

void file_pointer_main(const char *in_file_name, const char
*out_file_name);

void print_from_file(const char *file_name);
```

```
void create_or_append_file(const char *file_name);

int to_number(const char *file_name, int begin, int end);

void write_task_line(const char *in_file_name, FILE * out_file_ptr, int
begin, int end, int min);

void task(const char *in_file_name, const char *out_file_name);

bool is_spacer(int chr);

#endif //LAB1_FILEPOINTER_H
```

FilePointer.cpp

```
#include "FilePointer.h"

void file_pointer_main(const char *in_file_name, const char *out_file_name)
{
    create_or_append_file(in_file_name); // Prev editing input file (or
creating if not exists)
    std::cout << "Input file:\n";
    print_from_file(in_file_name); // printing input file
    task(in_file_name, out_file_name); // processing task
    std::cout << "Output file:\n";
    print_from_file(out_file_name); // printing output file
}

void print_from_file(const char *file_name) {

    FILE *file_ptr = fopen(file_name, "r");
    while (true) {
        wchar_t chr = fgetc(file_ptr);
        if (chr == -1) {
            std::wcout << '\n';
            break;
        }
        std::wcout << chr;
    }

    fclose(file_ptr);
}

void create_or_append_file(const char *file_name) {
    int mode = 0;
    FILE *file_ptr = fopen(file_name, "r");

    if (file_ptr != nullptr) { // if file found print content and ask what
to do
        std::cout << "Found file, content:\n";
        fclose(file_ptr);
        print_from_file(file_name);
```

```
        std::cout << "To rewrite file enter 0, to append 1, to not change 2:
";
        std::cin >> mode;
    }

    switch (mode) {
        case 0: {
            file_ptr = fopen(file_name, "w"); // rewrite mode
            break;
        }
        case 1: {
            file_ptr = fopen(file_name, "a"); // append mode
            break;
        }
        case 2: {
            return; // not edit
        }
        default: {
            throw;
        }
    }

    std::cout << "Writing to file.\nIf you want to end input, go to next
line, press Ctrl+B and then Enter.\n";

    int chr = getchar();
    while ((chr = getchar()) != 2)  // 2 is code of Ctrl+B
        fputc(chr, file_ptr);


    fclose(file_ptr);
}

int to_number(const char *file_name, int begin, int end) {
    FILE *file_ptr = fopen(file_name, "r");
    fseek(file_ptr, begin, SEEK_SET);

    int num = 0;

    for (int i = begin; i < end - 1; ++i) {
        int chr = fgetc(file_ptr);
        if (!isdigit(chr)) { // if one of chars is not digit it is word
            num = -1;
            break;
        }
        num = num * 10 + (chr - '0');
    }

    fclose(file_ptr);
    return num;
}
```

```
void write_task_line(const char *in_file_name, FILE* out_file_ptr, int
begin, int end, int min) {
    FILE *in_file_ptr = fopen(in_file_name, "r");

    fprintf(out_file_ptr, "%d ", min);

    int word_end = end;
    for (int i = end - 1; i >= begin - 1; --i) {
        fseek(in_file_ptr, i, SEEK_SET);
        int chr = fgetc(in_file_ptr);
        int num;

        if (is_spacer(chr)) {
            if (i < word_end - 1) {
                if ((num = to_number(in_file_name, i + 1, word_end + 1)) !=
-1) {
                    if (num != min)
                        fprintf(out_file_ptr, "%d ", num);
                }
            }
            word_end = i;
        }

        if (i == 0 && i < word_end){
            if ((num = to_number(in_file_name, i, word_end + 1)) != -1) {
                if (num != min)
                    fprintf(out_file_ptr, "%d ", num);
            }
        }
    }


    fseek(in_file_ptr, begin, SEEK_SET);
    int word_begin = begin;
    for (int i = begin; i < end; ++i) {
        int chr = fgetc(in_file_ptr);
        if (is_spacer(chr)) {
            if (i > word_begin) {
                if (to_number(in_file_name, word_begin, i + 1) == -1) {
                    fseek(in_file_ptr, word_begin, SEEK_SET);
                    for (int j = word_begin; j < i; ++j) {
                        fputc(fgetc(in_file_ptr), out_file_ptr);
                    }
                    fputc(' ', out_file_ptr);
                    fseek(in_file_ptr, i + 1, SEEK_SET);
                }
            }
            word_begin = i + 1;
        }

    }
```

```
    fseek(out_file_ptr, -1, SEEK_CUR);
    fputc('\n', out_file_ptr);

    fclose(in_file_ptr);
}


void task(const char *in_file_name, const char *out_file_name) {
    FILE *in_file_ptr = fopen(in_file_name, "r");
    FILE *out_file_ptr = fopen(out_file_name, "w");

    int line_begin = 0, word_begin = 0, cur_pos = 0, min, num;
    bool min_found = false, cont = true;

    while (cont) {
        int chr = fgetc(in_file_ptr);
        cur_pos++;
        if (is_spacer(chr)) {
            if (cur_pos > word_begin + 1) {
                if ((num = to_number(in_file_name, word_begin, cur_pos)) !=
-1) {
                    if (min_found) {
                        if (min > num)
                            min = num;
                    } else {
                        min = num;
                        min_found = true;
                    }
                }
            }
            switch (chr) {
                case -1: {
                    cont = false;
                }
                case 10: {
                    if (min_found) { // if min found write line to output
file
                        write_task_line(
                                in_file_name,
                                out_file_ptr,
                                line_begin,
                                cur_pos,
                                min
                        );
                    }
                    min_found = false;
                    line_begin = cur_pos;
                }
                default: {
                    word_begin = cur_pos;
                }
            }
        }
```

```
    }

    fclose(in_file_ptr);
    fclose(out_file_ptr);

}

bool is_spacer(int chr){
    return chr == -1 || chr == 10 || chr == 32;
}
```

Результати тестування

```
/home/okf0k/workspace/OP_labs/lab1/cmake-build-debug/lab1 -mode FilePointer
Ctrl+B char: 
Found file, content:
ord something 15 it is 50 13 56 five
червоний 36 крапка 45 75 сім
рядок без чисел

To rewrite file enter 0, to append 1, to not change 2: 1
Writing to file.
If you want to end input, go to next line, press Ctrl+B and then Enter.
27 Дописаний рядок 14

Input file:
ord something 15 it is 50 13 56 five
червоний 36 крапка 45 75 сім
рядок без чисел
27 Дописаний рядок 14

Output file:
13 56 50 15 ord something it is five
36 75 45 червоний крапка сім
14 27 Дописаний рядок
```

```
/home/okf0k/workspace/OP_labs/lab1/cmake-build-debug/lab1 -mode FileStream
Ctrl+B char: 
Found file, content:
ord something 15 it is 50 13 56 five
червоний 36 крапка 45 75 сім
рядок без чисел
27 Дописаний рядок 14
To rewrite file enter 0, to mode 1, to not change 2: 2
Input file:
ord something 15 it is 50 13 56 five
червоний 36 крапка 45 75 сім
рядок без чисел
27 Дописаний рядок 14
Output file:
13 56 50 15 ord something it is five
36 75 45 червоний крапка сім
14 Дописаний рядок

Process finished with exit code 0
```