Додаток 1

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1.2 з дисципліни «Основи програмування»

«Бінарні файли»

Варіант 12

Виконав студент: ІП-13 Дойчев Костянтин Миколайович

Перевірила: Вєчерковська Анастасія Сергіївна

Київ 2021

Лабораторна робота №1.2

Тема: Бінарні файли

Постановка задачі

12. Створити файл із списком справ на поточний день: умовна назва, час початку, передбачувана тривалість. Занести до нового файлу інформацію про справи, які потрібно виконати з 12.45 до 17.30. Вивести інформацію про вільний час (початок і кінець тимчасового проміжку та його тривалість).

Код:

Файл - main.cpp

```
#include <iostream>
#include <vector>
#include "utils.h"

using namespace std;

int main() {
    string initialFileName;
    cout << "Enter initial file name: ";
    cin >> initialFileName;

    Range RANGE = {"12:45", "17:30"};


    vector<Todo> initialTodos = {
            {"Buy milk",  "09:00", "00:15"},
            {"Buy eggs",  "12:45", "01:00"},
            {"Buy bread", "17:00", "00:45"},
    };

    cout << "Initial todos:" << endl;
    printTodos(initialTodos);

    cout << "Filling binary file..." << endl;
```

```
    fillBinaryFile(initialFileName, initialTodos);

    cout << "Reading binary file..." << endl;
    vector<Todo> todosFromFile = readBinaryFile(initialFileName,
initialTodos.size());

    cout << "Todos from " << initialFileName << ": " << endl;
    printTodos(todosFromFile);

    cout << "Filtering todos by range..." << endl;
    vector<Todo> filteredTodos = filterArray(todosFromFile, filterCallback,
RANGE);

    cout << "Todos in range : " << endl;
    printTodos(filteredTodos);

    string outputFilename;
    cout << "Enter output file name: ";
    cin >> outputFilename;

    cout << "Filling filtered todos in the  binary file..." << endl;
    fillBinaryFile(outputFilename, filteredTodos);
    cout << "Done!" << endl;

    cout << "Reading filtered todos from the binary file..." << endl;
    vector<Todo> todosFromOutputFile = readBinaryFile(outputFilename,
filteredTodos.size());

    cout << "Todos from " << outputFilename << ": " << endl;
    printTodos(todosFromOutputFile);

    vector<FreeTime> freeTime = getFreeTime(initialTodos);
    cout << "Free time: " << endl;
    printFreeTime(freeTime);

    return 0;
}
```

Файл: utils.cpp

```
#include "utils.h"


using namespace std;


vector<string> split(string text, const string &divider) {
    vector<string> output;
    size_t position = 0;
    string chunk;
    while ((position = text.find(divider)) != string::npos) {
        chunk = text.substr(0, position);
        output.push_back(chunk);
        text.erase(0, position + divider.length());
    }

    if (text.length() > 0) {
        output.push_back(text);
    }

    return output;
}


void printTodos(vector<Todo> todos) {
```

```
    for (auto todo: todos) {
        cout << "Title: " << todo.title << ", Begin time: " << todo.beginTime <<
", Duration: "
             << todo.duration
             << endl;
    }
}

void fillBinaryFile(const string &fileName, vector<Todo> todos) {
    ofstream file(fileName, ios::binary);
    if (!file.is_open()) {
        throw runtime_error("Could not open file " + fileName);
    }

    for (int i = 0; i < todos.size(); ++i) {
        file.write((char *) &todos[i], sizeof(Todo));
    }

    file.close();
    if (!file.good()) {
        throw runtime_error("Could not write to file " + fileName);
    }
}
```

```
vector<Todo> readBinaryFile(const string &fileName, int size) {
    vector<Todo> todos(size);
    ifstream file(fileName, ios::binary);
    if (!file.is_open()) {
        throw runtime_error("Could not open file " + fileName);
    }

    for (int i = 0; i < size; i++) {
        file.read((char *) &todos[i], sizeof(Todo));
    }

    file.close();
    if (!file.good()) {
        throw runtime_error("Could not read file " + fileName);
    }
    return todos;
}

vector<Todo> filterArray(vector<Todo> &arr, bool (*callback)(Todo, Range),
Range range) {
    vector<Todo> filteredArr;
    for (int i = 0; i < arr.size(); ++i) {
```

```
        if (callback(arr[i], range)) {
            filteredArr.push_back(arr[i]);
        }
    }
    return filteredArr;
}

int transformTimeToMilliseconds(const string &time) {
    vector<string> timeParts = split(time, ":");
    int hours = stoi(timeParts[0]);
    int minutes = stoi(timeParts[1]);
    return hours * 3600 * 1000 + minutes * 60 * 1000;
}

string transformMillisecondsToTime(int milliseconds) {
    int hours = milliseconds / (3600 * 1000);
    int minutes = (milliseconds % (3600 * 1000)) / (60 * 1000);
    return to_string(hours) + ":" + to_string(minutes);
}

string addTime(string time1, string time2) {
    int time1Milliseconds = transformTimeToMilliseconds(time1);
```

```
    int time2Milliseconds = transformTimeToMilliseconds(time2);

    int sumMilliseconds = time1Milliseconds + time2Milliseconds;

    return transformMillisecondsToTime(sumMilliseconds);

}


string subtractTime(string time1, string time2) {

    int time1Milliseconds = transformTimeToMilliseconds(time1);

    int time2Milliseconds = transformTimeToMilliseconds(time2);

    int differenceMilliseconds = time1Milliseconds - time2Milliseconds;

    return transformMillisecondsToTime(differenceMilliseconds);

}


bool filterCallback(Todo todo, Range range) {

    string timeSum = addTime(todo.beginTime, todo.duration);

    int timeSumMilliseconds = transformTimeToMilliseconds(timeSum);

    int minMilliseconds = transformTimeToMilliseconds(range.min);

    int maxMilliseconds = transformTimeToMilliseconds(range.max);

    return timeSumMilliseconds >= minMilliseconds && timeSumMilliseconds <=
maxMilliseconds;

};


bool sortByBeginTime(Todo todo1, Todo todo2) {

    return transformTimeToMilliseconds(todo1.beginTime) <
transformTimeToMilliseconds(todo2.beginTime);
```

```
}


vector<FreeTime> getFreeTime(vector<Todo> todos, string startTime, string
endTime) {
    vector<FreeTime> freeTime;
    string cursor = startTime;

    sort(todos.begin(), todos.end(), sortByBeginTime);

    for (auto todo: todos) {
        string todoEndTime = addTime(todo.beginTime, todo.duration);
        int cursorMilliseconds = transformTimeToMilliseconds(cursor);
        int todoBeginMilliseconds = transformTimeToMilliseconds(todo.beginTime);
        int todoEndMilliseconds = transformTimeToMilliseconds(todoEndTime);

        if (cursorMilliseconds < todoBeginMilliseconds < todoEndMilliseconds) {
            FreeTime freeTimeItem = {cursor, todo.beginTime};
            freeTime.push_back(freeTimeItem);
            cursor = todoEndTime;
        } else if (cursorMilliseconds < todoBeginMilliseconds) {
            cursor = todoEndTime;
        }
    }
```

```
    if (transformTimeToMilliseconds(cursor) <
transformTimeToMilliseconds(endTime)) {

        FreeTime freeTimeItem = {cursor, endTime};

        freeTime.push_back(freeTimeItem);

    }



    return freeTime;

}



void printFreeTime(std::vector<FreeTime> freeTime) {

    for (auto freeTimeItem: freeTime) {

        string duration = subtractTime(freeTimeItem.endTime,
freeTimeItem.beginTime);

        cout << "Begin time: " << freeTimeItem.beginTime << ", End time: " <<
freeTimeItem.endTime << " , Duration: "

             << duration << endl;

    }

}
```

Файл:utils.h

```
#ifndef LAB_1_2_UTILS_H
#define LAB_1_2_UTILS_H

#endif //LAB_1_2_UTILS_H

#include <iostream>
```

```cpp
#include <vector>
#include <string>
#include <fstream>

struct Todo {
    std::string title;
    std::string beginTime;
    std::string duration;
};

struct FreeTime {
    std::string beginTime;
    std::string endTime;
};

struct Range {
    std::string min;
    std::string max;
};

std::vector<std::string> split(std::string text, const std::string &divider);

void printTodos(std::vector<Todo> todos);

void fillBinaryFile(const std::string &fileName,std::vector<Todo> todos);

std::vector<Todo> readBinaryFile(const std::string &fileName, int size);

std::vector<Todo> filterArray(std::vector<Todo> &arr, bool (*callback)(Todo,
Range), Range range);

bool filterCallback(Todo todo, Range range);

int transformTimeToMilliseconds(const std::string &time);

std::string transformMillisecondsToTime(int milliseconds);

std::string addTime(std::string time1, std::string time2);

std::string subtractTime(std::string time1, std::string time2);

bool sortByBeginTime(Todo todo1, Todo todo2);

std::vector<FreeTime> getFreeTime(std::vector<Todo> todos, std::string
startTime = "00:00", std::string endTime="23:59");

void printFreeTime(std::vector<FreeTime> freeTime);
```

Дані і консоль:

```
/Users/Kostia/Documents/Programming/kpi/programming-basics/C-plus-plus/bachelor/year-1/semester-2/lab-1.2/cmake-build-debug/lab_1_2
Enter initial file name: init.bin
Initial todos:
Title: Buy milk, Begin time: 09:00, Duration: 00:15
Title: Buy eggs, Begin time: 12:45, Duration: 01:00
Title: Buy bread, Begin time: 17:00, Duration: 00:45
Filling binary file...
Reading binary file...
Todos from init.bin:
Title: Buy milk, Begin time: 09:00, Duration: 00:15
Title: Buy eggs, Begin time: 12:45, Duration: 01:00
Title: Buy bread, Begin time: 17:00, Duration: 00:45
Filtering todos by range...
Todos in range :
Title: Buy eggs, Begin time: 12:45, Duration: 01:00
Enter output file name: output.bin\
Filling filtered todos in the  binary file...
Done!
Reading filtered todos from the binary file...
Todos from output.bin\:
Title: Buy eggs, Begin time: 12:45, Duration: 01:00
Free time:
Begin time: 00:00, End time: 09:00 , Duration: 9:0
Begin time: 9:15, End time: 12:45 , Duration: 3:30
Begin time: 13:45, End time: 17:00 , Duration: 3:15
Begin time: 17:45, End time: 23:59 , Duration: 6:14
```