Додаток А

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

ЗВІТ
з виконання лабораторної роботи № 2
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

9. Створити файл із списком пацієнтів, записаних на прийом до лікаря: прізвище пацієнта, дата попереднього відвідування лікаря та час, на який пацієнт записаний. Видалити з файлу записи про пацієнтів, час прийому яких минув. Створити два нових файли: в один занести відомості про вторинних хворих (попереднє відвідування яких було протягом 10-ти останніх днів), а в другий – про решту пацієнтів.

Текст програми

main.cpp

```cpp
#include <iostream>
#include "ClinicAttendance.h"


const char INPUT_FILE_NAME[] = "in.bin";
const char SECONDARY_PATIENTS_FILE_NAME[] = "secondary_patients.bin";
const char OTHER_PATIENTS_FILE_NAME[] = "other_patients.bin";


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

   create_or_append_file(INPUT_FILE_NAME);

   std::cout << "Input file:\n";
   print_from_file(INPUT_FILE_NAME);

   delete_by_filter(INPUT_FILE_NAME);

   std::cout << "Input file after deleting:\n";
   print_from_file(INPUT_FILE_NAME);

   sort_patients(
           INPUT_FILE_NAME,
           SECONDARY_PATIENTS_FILE_NAME,
           OTHER_PATIENTS_FILE_NAME
   );


   std::cout << "Secondary patients file:\n";
   print_from_file(SECONDARY_PATIENTS_FILE_NAME);

   std::cout << "Other patients file:\n";
   print_from_file(OTHER_PATIENTS_FILE_NAME);

}
```

ClinicAttendance.h

```cpp
#ifndef LAB2_CLINICATTENDANCE_H
#define LAB2_CLINICATTENDANCE_H

#include <string>
#include <fstream>
#include <iostream>
#include <ctime>

const char TEMP_FILE_NAME[] = "temp.bin";

struct Date {
   int day;
   int month;
   int year;
};

struct Time {
   int minutes;
   int hours;
};

struct ClinicAttendance {
   std::string surname;
   Date prev_attendance_date;
   Time cur_attendance_time;
};

tm cur_time();

tm prev_attendance_time(ClinicAttendance const &ca);

void create_or_append_file(const char file_name[]);

void write_ca_record(std::ostream &out, ClinicAttendance const &data);

ClinicAttendance *read_ca_record(std::istream &in);

void write_console_to_file(const char file_name[], bool append = false);

void print_from_file(const char file_name[]);

bool attendance_passed_filter(const ClinicAttendance &);

bool secondary_patient_filter(const ClinicAttendance &);

void delete_by_filter(
       const char file_name[],
       bool (*filter)(const ClinicAttendance &) = attendance_passed_filter
);

void sort_patients(
       const char in_file_name[],
        const char passed_file_name[],
        const char not_passed_file_name[],
        bool (*filter)(const ClinicAttendance &) = secondary_patient_filter
);

bool validate_ca_record(ClinicAttendance const &ca);


#endif //LAB2_CLINICATTENDANCE_H
```

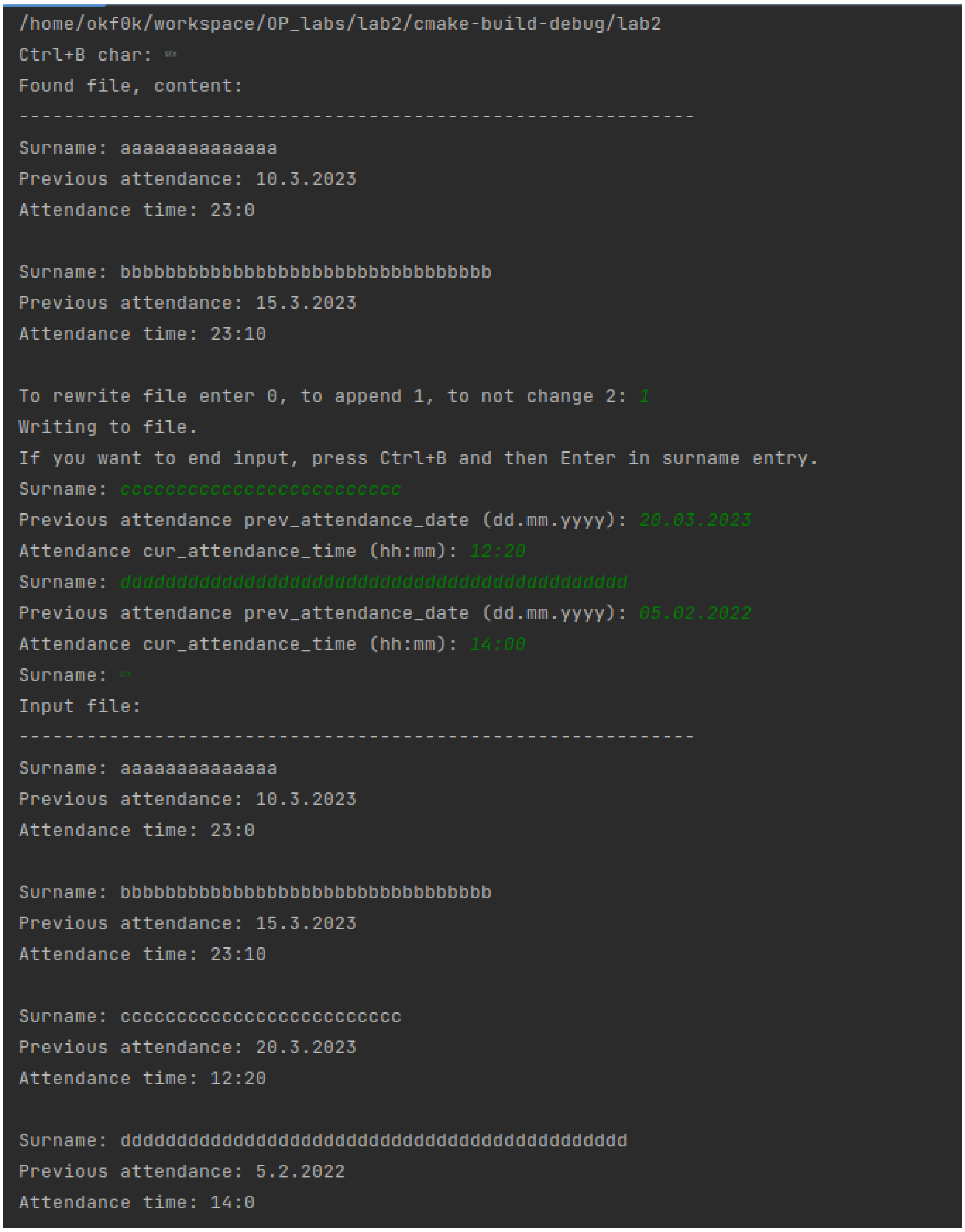
ClinicAttendance.cpp

```
#include "ClinicAttendance.h"

void create_or_append_file(const char file_name[]) {
    std::ifstream in_file(file_name, std::ios::binary);
    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 append 1, to not change 2:
";
        std::cin >> mode;
    }

    switch (mode) {
        case 0:
        case 1:
            write_console_to_file(file_name, mode);
            break;
        case 2:
            return; // not edit
        default:
            throw;
    }

}

void write_ca_record(std::ostream &out, ClinicAttendance const &data) {
    size_t size = data.surname.size();
    out.write(reinterpret_cast<const char *>(&size), sizeof(size));
    out.write(&data.surname[0], size);
    out.write(reinterpret_cast<char const *>(&data.prev_attendance_date),
sizeof(data.prev_attendance_date));
    out.write(reinterpret_cast<char const *>(&data.cur_attendance_time),
sizeof(data.cur_attendance_time));

}

ClinicAttendance *read_ca_record(std::istream &in) {
    auto ca = new ClinicAttendance;
    size_t size;
    in.read(reinterpret_cast<char *>(&size), sizeof(size));

    if (in.eof())
        return nullptr;

    ca->surname.resize(size);
    in.read(&ca->surname[0], size);

    in.read(reinterpret_cast<char *>(&ca->prev_attendance_date),
sizeof(ca->prev_attendance_date));
    in.read(reinterpret_cast<char *>(&ca->cur_attendance_time),
sizeof(ca->cur_attendance_time));


    return ca;
}

void write_console_to_file(const char file_name[], bool append) {
    ClinicAttendance ca;

    std::ofstream file;
    if (append)
        file.open(file_name, std::ios::binary | std::ios::app);
    else
        file.open(file_name, std::ios::binary);

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

    while (true) {
        std::cout << "Surname: ";
        std::cin >> ca.surname;
        if (ca.surname[0] == 2) // 2 is code of Ctrl+B
            break;
        std::cout << "Previous attendance prev_attendance_date (dd.mm.yyyy):
";
        scanf("%i.%i.%i",
              &ca.prev_attendance_date.day,
              &ca.prev_attendance_date.month,
              &ca.prev_attendance_date.year
        );
        std::cout << "Attendance cur_attendance_time (hh:mm): ";
        scanf("%i:%i",
              &ca.cur_attendance_time.hours,
              &ca.cur_attendance_time.minutes
        );
        if (validate_ca_record(ca))
            write_ca_record(file, ca);
        else
            std::cout << "Validation error!!!\nTry one more time\n";
    }

    file.close();
 }

 void print_from_file(const char file_name[]) {
    ClinicAttendance *ca;
    std::ifstream file(file_name, std::ios::binary);

    if (!file) {
        std::cout << "Error opening file. Program aborting.\n";
        throw;
    }

    std::cout <<
 "------------------------------------------------------------\n";
    while ((ca = read_ca_record(file))) {
        std::cout << "Surname: " << ca->surname << '\n'
                  << "Previous attendance: "
                  << ca->prev_attendance_date.day << "."
                  << ca->prev_attendance_date.month << "."
                  << ca->prev_attendance_date.year << '\n'
                  << "Attendance time: "
                  << ca->cur_attendance_time.hours << ':'
                  << ca->cur_attendance_time.minutes << "\n\n";

    }

    file.close();
 }

 bool attendance_passed_filter(const ClinicAttendance &ca) {
    tm now = cur_time();
    return (now.tm_hour * 60 + now.tm_min) >
           (ca.cur_attendance_time.hours * 60 +
 ca.cur_attendance_time.minutes);
 }

 bool secondary_patient_filter(const ClinicAttendance &ca) {
    tm now = cur_time(), time = prev_attendance_time(ca);
    return difftime(mktime(&now), mktime(&time)) <= 10 * 24 * 3600;

 }

 tm cur_time() {
    time_t t = std::time(nullptr);
    return *std::localtime(&t);
 }

 tm prev_attendance_time(ClinicAttendance const &ca) {
    tm time = cur_time();

    time.tm_year = ca.prev_attendance_date.year - 1900;
```

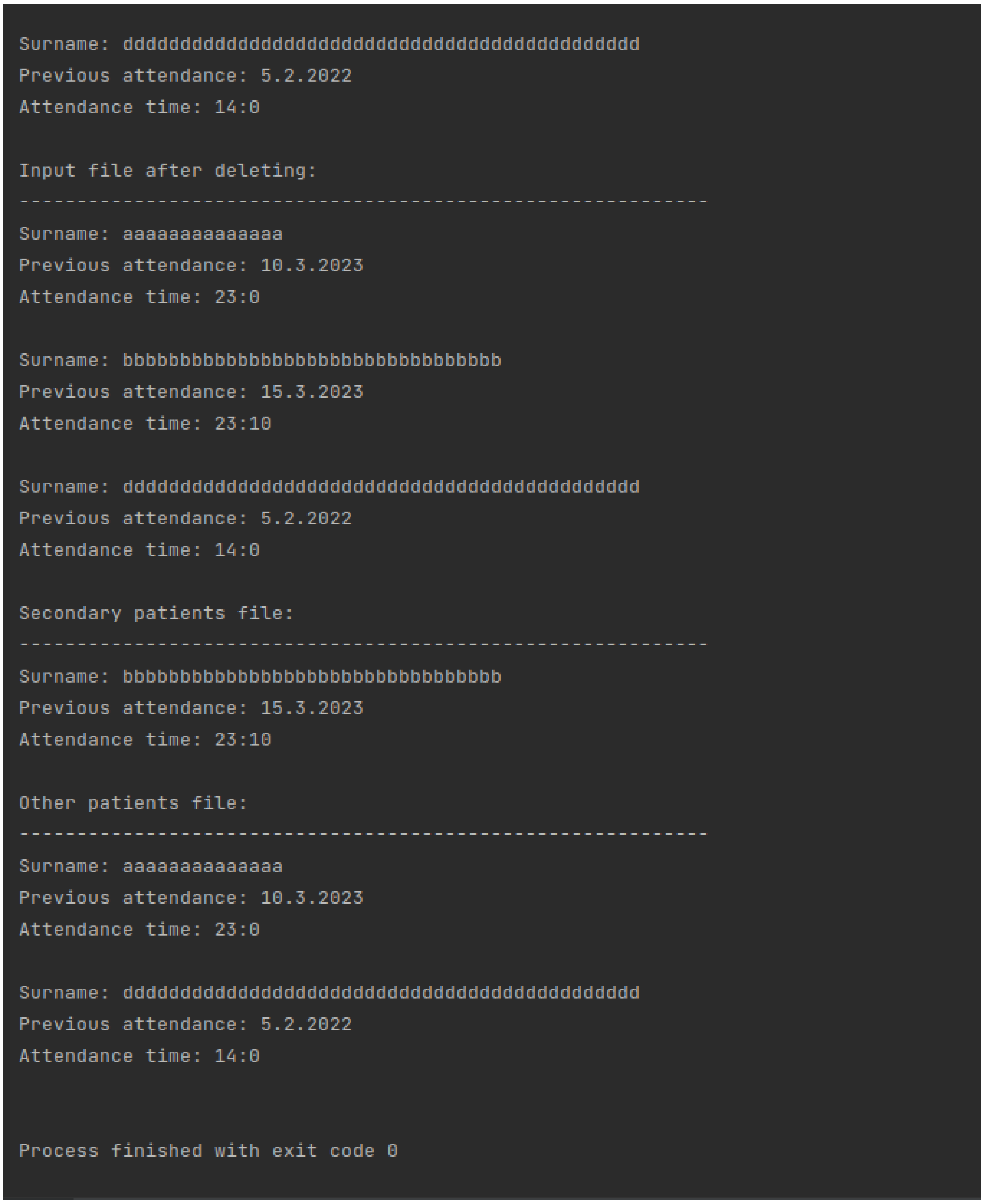
```
    time.tm_mon = ca.prev_attendance_date.month - 1;
    time.tm_mday = ca.prev_attendance_date.day;
    return time;
}


void delete_by_filter(const char file_name[], bool (*filter)(const
ClinicAttendance &)) {
    std::ifstream file(file_name, std::ios::binary);
    std::ofstream temp_file(TEMP_FILE_NAME, std::ios::binary);

    ClinicAttendance *ca;

    while ((ca = read_ca_record(file))) {
        if (!filter(*ca)) {
            write_ca_record(temp_file, *ca);
        }
    }

    file.close();
    temp_file.close();
    remove(file_name);
    rename(TEMP_FILE_NAME, file_name);


}


void sort_patients(
        const char in_file_name[],
        const char passed_file_name[],
        const char not_passed_file_name[],
        bool (*filter)(const ClinicAttendance &)
) {
    std::ifstream in_file(in_file_name, std::ios::binary);
    std::ofstream pass_file(passed_file_name, std::ios::binary);
    std::ofstream not_pass_file(not_passed_file_name, std::ios::binary);

    ClinicAttendance *ca;

    while ((ca = read_ca_record(in_file))) {
        write_ca_record(
                filter(*ca) ? pass_file : not_pass_file,
                *ca
        );
    }

    in_file.close();
    pass_file.close();
    not_pass_file.close();

}
```

```
bool validate_ca_record(ClinicAttendance const &ca) {
   if (!(0 <= ca.cur_attendance_time.hours && ca.cur_attendance_time.hours
< 24))
       return false;
   if (!(0 <= ca.cur_attendance_time.minutes &&
ca.cur_attendance_time.minutes < 60))
       return false;

   if (!(1 <= ca.prev_attendance_date.day && ca.prev_attendance_date.day <=
31))
       return false;
   if (!(1 <= ca.prev_attendance_date.month &&
ca.prev_attendance_date.month <= 12))
       return false;
   if (2000 > ca.prev_attendance_date.year)
       return false;


   tm now = cur_time(), time = prev_attendance_time(ca);
   return difftime(mktime(&now), mktime(&time)) > 0;

}
```

Результати тестування



```
/home/okf0k/workspace/OP_labs/lab2/cmake-build-debug/lab2
Ctrl+B char: ␂
Found file, content:
------------------------------------------------------------
Surname: aaaaaaaaaaaaaa
Previous attendance: 10.3.2023
Attendance time: 23:0

Surname: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
Previous attendance: 15.3.2023
Attendance time: 23:10

To rewrite file enter 0, to append 1, to not change 2: 1
Writing to file.
If you want to end input, press Ctrl+B and then Enter in surname entry.
Surname: ccccccccccccccccccccccccc
Previous attendance prev_attendance_date (dd.mm.yyyy): 20.03.2023
Attendance cur_attendance_time (hh:mm): 12:20
Surname: ddddddddddddddddddddddddddddddddddddddddddddd
Previous attendance prev_attendance_date (dd.mm.yyyy): 05.02.2022
Attendance cur_attendance_time (hh:mm): 14:00
Surname: ␂
Input file:
------------------------------------------------------------
Surname: aaaaaaaaaaaaaa
Previous attendance: 10.3.2023
Attendance time: 23:0

Surname: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
Previous attendance: 15.3.2023
Attendance time: 23:10

Surname: ccccccccccccccccccccccccc
Previous attendance: 20.3.2023
Attendance time: 12:20

Surname: ddddddddddddddddddddddddddddddddddddddddddddd
Previous attendance: 5.2.2022
Attendance time: 14:0
```



```
Surname: ddddddddddddddddddddddddddddddddddddddddddddd
Previous attendance: 5.2.2022
Attendance time: 14:0

Input file after deleting:
------------------------------------------------------------
Surname: aaaaaaaaaaaaaa
Previous attendance: 10.3.2023
Attendance time: 23:0

Surname: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
Previous attendance: 15.3.2023
Attendance time: 23:10

Surname: ddddddddddddddddddddddddddddddddddddddddddddd
Previous attendance: 5.2.2022
Attendance time: 14:0

Secondary patients file:
------------------------------------------------------------
Surname: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
Previous attendance: 15.3.2023
Attendance time: 23:10

Other patients file:
------------------------------------------------------------
Surname: aaaaaaaaaaaaaa
Previous attendance: 10.3.2023
Attendance time: 23:0

Surname: ddddddddddddddddddddddddddddddddddddddddddddd
Previous attendance: 5.2.2022
Attendance time: 14:0


Process finished with exit code 0
```