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

ЗВІТ

з виконання лабораторної роботи № 2
з кредитного модуля
“Основи програмування-2. Методології програмування”

Варіант № 21

Виконав:
студент 1-го курсу
гр. ІП-22 ФІОТ
Патріюк Юрій Олексійович

Київ 2023

ПОСТАНОВКА ЗАДАЧІ

21. Створити файл із списком справ на поточний день: умовна назва, час початку, передбачувана тривалість. Визначити, яка справа за списком наступна (найближча до поточного часу). Створити файл з інформацією про вільний час у другій половині дня (після 13:00): початок та закінчення тимчасового проміжку та його тривалість (розрахувати).

ТЕКСТ ПРОГРАМИ

Вміст файлу main.cpp

```
#include "MainFunctions.h"

int main() {

    string name = "exercises.dat";
    string result = "free_time.dat";
    WriteInFile(name);
    AppendInFile(name);
    ShowFile(name);
    FindClosestExercise(name);
    FindFreeTime(name,result);
    ShowFreeTime(result);

    return 0;
}
```

Вміст файлу MainFunctions.h

```
#include <iostream>
#include <fstream>
#include <cstring>
#include "Exercise.h"
using namespace std;

void WriteInFile(const string&);
void AppendInFile(const string&);
void ShowFile(const string&);
void FindClosestExercise(const string&);
void FindFreeTime(const string&, const string&);
void ShowFreeTime(const string&);
```

Вміст файлу MainFunctions.cpp

```
#include "MainFunctions.h"
#include <ctime>

//допоміжна функція введення та перевірки значень
bool CheckAndSetEnteredTime(int buf[])
{
    int h,m;
    Time t;
    cout<<"Enter hours:";
    cin>>h;
```

```
    cout<<"Enter minutes:";
    cin>>m;
    if(t.CheckHours(h)&&t.CheckMinutes(m))
    {
        buf[0] = h;
        buf[1] = m;
        return true;
    }
    else {
        cout<<"Error! Incorrect time!"<<endl;
        return false;
    }
}
//сортування масиву
void BubbleSort(Exercise array[], int size)
{
    Exercise temp;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size-i-1; ++j) {
            if(array[j].GetTime().GetHours() > array[j+1].GetTime().GetHours() ||
(array[j].GetTime().GetHours() == array[j+1].GetTime().GetHours() &&
array[j].GetTime().GetMinutes() > array[j+1].GetTime().GetMinutes()))
            {
                temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
            }

        }
    }
}
//коректування годин та хвилин
void CorrectTime(int &h, int &m)
{
    while (m >= 60) {
        m -= 60;
        h++;
    }
}

//запис у файл елементів класу Exercise
void WriteInFile(const string& main_file)
{
    ofstream outFile(main_file,ios::binary);
    Exercise ex;
    char check = 0;
    int temp[2];
    while(check !='n') {
        cout<<"Enter name of exercise: ";
        char* line;
        line = (char*)calloc(40, sizeof(char));
        gets(line);
        ex.SetName(line);

        do {
            cout << "Enter time of start of exercise"<<endl;
        }while(!CheckAndSetEnteredTime(temp));
        ex.SetTime(temp[0],temp[1]);
        do {
            cout<<"Enter duration of exercise"<<endl;
        }while(!CheckAndSetEnteredTime(temp));
        ex.SetDuration(temp[0],temp[1]);

        outFile.write((char*)&ex, sizeof(Exercise));
```

```
        do {
            cout<<endl;
            cout << "Do you want to continue? y/n: ";
            cin >> check;
        }while(check!='y' && check!='n');
        fflush(stdin);
    }
    outFile.close();

}
//вставлення в кінець файлу елементів класу Exercise
void AppendInFile(const string& main_file)
{
    char answer=' ';
    cout<<"Do you want to append? y/n: ";
    cin>>answer;
    fflush(stdin);
    if(answer=='y') {
        ofstream outFile(main_file, ios::app | ios::binary);
        Exercise ex;
        char check = 0;
        int temp[2];
        while (check != 'n') {
            cout << "Enter name of exercise: ";
            char *line;
            line = (char *) calloc(40, sizeof(char));
            gets(line);
            ex.SetName(line);

            do {
                cout << "Enter time of start of exercise" << endl;
            } while (!CheckAndSetEnteredTime(temp));
            ex.SetTime(temp[0], temp[1]);
            do {
                cout << "Enter duration of exercise" << endl;
            } while (!CheckAndSetEnteredTime(temp));
            ex.SetDuration(temp[0], temp[1]);

            outFile.write((char *) &ex, sizeof(Exercise));

            do {
                cout << endl;
                cout << "Do you want to continue? y/n: ";
                cin >> check;
            } while (check != 'y' && check != 'n');
            cout<<endl;
            fflush(stdin);
        }
        outFile.close();
    }
}
//вивід елементів класу Exercise, що містяться у файлі
void ShowFile(const string& main_file)
{
    ifstream iFile(main_file,ios::binary);
    cout<<"Content of "<<main_file<<endl;
    Exercise ex;
    iFile.read((char*)&ex,sizeof(ex));
    do{
        ex.ShowExercise();
        cout<<endl;
        iFile.read((char*)&ex,sizeof(ex));
    }while(!iFile.eof());
```

```
    iFile.close();
}
//пошук найближчого до поточного часу завдання
void FindClosestExercise(const string& main_file)
{
    ifstream iFile(main_file, ios::binary);
    Exercise exercise;
    bool is_found = false;
    time_t curr_time = time(nullptr);
    const tm calendar_time = *localtime(addressof(curr_time));
    while(!iFile.eof() && !is_found)
    {
        iFile.read((char*)&exercise,sizeof(exercise));
        if(calendar_time.tm_hour<exercise.GetTime().GetHours())
            is_found = true;
        else if(calendar_time.tm_hour == exercise.GetTime().GetHours()) {
            if (calendar_time.tm_min < exercise.GetTime().GetMinutes())
                is_found = true;
        }
    }
    if(is_found)
    {
        cout<<"Closest exercise: "<<endl;
        exercise.ShowExercise();
    }
    else
        cout<<"Closest wasn't found"<<endl;
    iFile.close();
    cout<<endl;

}
//пошук запис у файл вільного часу
void FindFreeTime(const string& main_file, const string& result_file)
{
    ifstream iFile(main_file, ios::binary);
    iFile.seekg(0,ios::end);
    int amount_of_all = iFile.tellg()/sizeof(Exercise);
    iFile.seekg(0,ios::beg);


    Exercise ex;
    int complete_amount=0;
    Exercise* exercises = (Exercise*) malloc(sizeof(Exercise));
    for (int i = 0; i < amount_of_all; ++i) {
        iFile.read((char*)&ex, sizeof(Exercise));
        if(ex.GetTime().GetHours()>=13) {
            complete_amount++;
            Exercise* temp = new Exercise[complete_amount];
            if(complete_amount>1)
                copy(exercises,exercises+complete_amount-1,temp);
            delete[] exercises;
            exercises = temp;
            exercises[complete_amount-1] = ex;
        }
    }
    iFile.close();
    BubbleSort(exercises,complete_amount);

    ofstream oFile(result_file,ios::binary);

    int i = 0, hour_end, minutes_end, hour_start, minutes_start;
    Time start, end, duration;
    if(exercises[i].GetTime().GetHours()>13 || exercises[i].GetTime().GetMinutes()>0)
    {
```

```
        start.SetTime(13,0);

end.SetTime(exercises[i].GetTime().GetHours(),exercises[i].GetTime().GetMinutes());
        duration.SetTime(exercises[i].GetTime().GetHours()-13,
exercises[i].GetTime().GetMinutes());
        oFile.write((char*)&start,sizeof(Time));
        oFile.write((char*)&end,sizeof(Time));
        oFile.write((char*)&duration,sizeof(Time));
    }

    while(i<complete_amount-1)
    {
        hour_end = exercises[i].GetTime().GetHours() +
exercises[i].GetDuration().GetHours();
        minutes_end = exercises[i].GetTime().GetMinutes() +
exercises[i].GetDuration().GetMinutes();
        hour_start = exercises[i+1].GetTime().GetHours();
        minutes_start = exercises[i+1].GetTime().GetMinutes();
        CorrectTime(hour_end,minutes_end);

        if(hour_start>hour_end || (hour_start==hour_end &&
minutes_start>minutes_end))
        {
            start.SetTime(hour_end,minutes_end);
            end.SetTime(hour_start,minutes_start);
            if(minutes_start-minutes_end<0) {
                duration.SetTime(hour_start - hour_end - 1,minutes_start -
minutes_end + 60);
            }
            else {
                duration.SetTime(hour_start - hour_end, minutes_start - minutes_end);
            }
            oFile.write((char*)&start,sizeof(Time));
            oFile.write((char*)&end,sizeof(Time));
            oFile.write((char*)&duration,sizeof(Time));
        }
        i++;
    }
    int end_hour =
exercises[i].GetTime().GetHours()+exercises[i].GetDuration().GetHours();
    int end_minutes =
exercises[i].GetTime().GetMinutes()+exercises[i].GetDuration().GetMinutes();
    CorrectTime(end_hour,end_minutes);
    if(end_hour<23 || end_minutes<59)
    {

        start.SetTime(end_hour,end_minutes);
        end.SetTime(23,59);
        duration.SetTime(23-end_hour,59-end_minutes);
        oFile.write((char*)&start,sizeof(Time));
        oFile.write((char*)&end,sizeof(Time));
        oFile.write((char*)&duration,sizeof(Time));
    }

    oFile.close();
}
//виведення вільного часу
void ShowFreeTime(const string& main_file)
{
    ifstream iFile(main_file,ios::binary);
    Time start, end, duration;
    iFile.read((char *) &start, sizeof(Time));
    iFile.read((char *) &end, sizeof(Time));
    iFile.read((char *) &duration, sizeof(Time));
```

```
    cout<<"Free time: "<<endl;
    do {
        cout << start.GetHours() << ':';
        if (start.GetMinutes() < 10)
            cout << 0;
        cout << start.GetMinutes() << '-' << end.GetHours() << ':';
        if (end.GetMinutes() < 10)
            cout << 0;
        cout << end.GetMinutes() << endl;
        cout << "Duration: ";
        if (duration.GetHours() < 10)
            cout << 0;
        cout << duration.GetHours() << ':';
        if (duration.GetMinutes() < 10)
            cout << 0;
        cout << duration.GetMinutes() << endl;
        iFile.read((char *) &start, sizeof(Time));
        iFile.read((char *) &end, sizeof(Time));
        iFile.read((char *) &duration, sizeof(Time));
    }while(!iFile.eof());

    iFile.close();
}
```

Вміст файлу Exercise.h

```
#include "Time.h"
#include <cstring>
#define MAX_SIZE 256

struct Exercise{
private:
    char name[MAX_SIZE];
    Time time;
    Time duration;
public:
    Time GetTime() const;
    Time GetDuration() const;
    char* GetName();
    void SetTime(int, int);
    void SetDuration(int, int);
    void SetName(const char* n);
    Exercise();
    Exercise(Exercise&);
    void ShowExercise();
};
```

Вміст файлу Exercise.cpp

```
#include "Exercise.h"

Time Exercise::GetTime() const
{
    return time;
}
Time Exercise::GetDuration() const
{
    return duration;
}
char* Exercise::GetName() {
    return name;
```

```
}
void Exercise::SetTime(int h, int m)
{
    time.SetHours(h);
    time.SetMinutes(m);
}
void Exercise::SetDuration(int h, int m)
{
    duration.SetHours(h);
    duration.SetMinutes(m);
}
void Exercise::SetName(const char* n)
{
    strcpy(name,n);
}
Exercise::Exercise()
{
    time = Time();
    duration = Time();
}
Exercise::Exercise(Exercise& obj)
{
    strcpy(name,obj.name);
    time = obj.time;
    duration = obj.duration;
}

void Exercise::ShowExercise()
{
    cout<<name<<endl;
    cout<<"Time: ";time.ShowTime();
    cout<<"Duration: ";duration.ShowTime();
}
```

Вміст файлу Time.h

```
#include <iostream>
using namespace std;

struct Time {
private:
    int hour;
    int min;
public:
    Time();
    Time(int, int);
    int GetHours() const;
    int GetMinutes() const;
    void SetHours(int h);
    void SetMinutes(int m);
    void SetTime(int h, int m);
    void ShowTime() const;
    bool CheckHours(int) const;
    bool CheckMinutes(int) const;

};
```

Вміст файлу Time.cpp

```
#include "Time.h"

Time::Time() {
    hour= 0;
    min = 0;
}
Time::Time(int hour, int min)
{
    if(CheckHours(hour) && CheckMinutes(min))
    {
        this->hour=hour;
        this->min=min;
    }
    else
    {
        cout<<"Incorrect time!"<<endl;
        this->hour=13;
        this->min=0;
    }
}
bool Time::CheckHours(int hours) const{
    if(hours>=24 || hours<0)
        return false;
    else
        return true;
}
bool Time::CheckMinutes(int minutes) const{
    if(minutes>60 || minutes<0)
        return false;
    else
        return true;
}
int Time::GetHours() const
{
    return hour;
}
int Time::GetMinutes() const
{
    return min;
}
void Time::SetHours(int h)
{
    hour = h;
}
void Time::SetMinutes(int m) {
    min = m;
}
void Time::SetTime(int h, int m)
{
    hour = h;
    min = m;
}

void Time::ShowTime() const{
    if(this->hour<10)
        cout<<0;
    cout<<hour<<':';
    if(this->min<10)
        cout<<0;
    cout<<min<<endl;
}
```

РЕЗУЛЬТАТ ТЕСТУВАННЯ

```
Enter name of exercise:do homework
 Enter time of start of exercise
Enter hours:20
Enter minutes:15
Enter duration of exercise
Enter hours:1
Enter minutes:50

Do you want to continue? y/n:n
 Do you want to append? y/n:y
 Enter name of exercise:prepare lunch
 Enter time of start of exercise
Enter hours:16
Enter minutes:45
Enter duration of exercise
Enter hours:0
Enter minutes:45

Do you want to continue? y/n:n

Content of exercises.dat
do homework
Time: 20:15
Duration: 01:50

prepare lunch
Time: 16:45
Duration: 00:45

Closest exercise:
do homework
Time: 20:15
Duration: 01:50

Free time:
13:00-16:45
Duration: 03:45
17:30-20:15
Duration: 02:45
22:05-23:59
Duration: 01:54

Process finished with exit code 0
```

```
Enter name of exercise:prepare dinner
 Enter time of start of exercise
Enter hours:14
Enter minutes:15
Enter duration of exercise
Enter hours:1
Enter minutes:15

Do you want to continue? y/n:y
 Enter name of exercise:laundry
 Enter time of start of exercise
Enter hours:17
Enter minutes:45
Enter duration of exercise
Enter hours:1
Enter minutes:10

Do you want to continue? y/n:n
 Do you want to append? y/n:y
 Enter name of exercise:homework
 Enter time of start of exercise
Enter hours:19
Enter minutes:20
Enter duration of exercise
Enter hours:2
Enter minutes:30

Do you want to continue? y/n:n

Content of exercises.dat
prepare dinner
Time: 14:15
Duration: 01:15

laundry
Time: 17:45
Duration: 01:10

homework
Time: 19:20
Duration: 02:30

Closest exercise:
prepare dinner
Time: 14:15
Duration: 01:15

Free time:
13:00-14:15
Duration: 01:15
15:30-17:45
Duration: 02:15
18:55-19:20
Duration: 00:25
21:50-23:59
Duration: 02:09

Process finished with exit code 0
```

Посилання на GitHub: https://github.com/YuriiPatriuk12/Labs.git