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 № 2 з дисципліни
«Основи програмування 2.
Модульне програмування»

«Файли даних. Бінарні файли»

Варіант 22

Виконав студент _____ІП-15_Мєшков_Андрій_Ігорович_____

(шифр, прізвище, ім'я, по батькові)

Перевірив _____Вечерковська Анастасія Сергіївна_____
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 2
Файли даних. Бінарні файли

Мета – вивчити особливості створення і обробки бінарних файлів.

Варіант 22

Завдання.

22. 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).

Код C++
Lab2.cpp

```cpp
#include <iostream>
#include <ctime>
#include "function.hpp"
#include <fstream>
#include <string>
#include <vector>
using namespace std;


int main()
{


    string name_in = to_do_list();//створення вихідного файлу
    cout<<endl;
    output_first(name_in);//вивід вихідного файлу

    vector <EList> text = file_to_text(name_in);//перенесення тексту файлу у змінну

    next_event(text);//знаходження наступної події

    string name_out = free_time(text);//створення другого файлу
    output_second(name_out);//вивід створеного файлу
    cout<<endl;
    cin.get();


}

```

function.hpp

```cpp
#ifndef function_hpp
#define function_hpp
#include <string>
using namespace std;
//структура для запису часу
struct Time {
    int hours, minutes;

    Time() {}

    Time(string time_str) {
        this->hours = atoi(time_str.substr(0, 2).c_str());
        this->minutes = atoi(time_str.substr(3, 2).c_str());
    }
};
//структура для данних події
struct EList {
    string name;
    Time time_start, duration;
};

//структура для вільного часу
struct FreeList{
    Time start, finish, free_dur;
};


string to_do_list();
string verifyName(string);
string verifyTime(string, string, string);
string verifyDuration(string, string);
bool hours(string);
bool dur(string, string, string);
bool duration_normal(string, string);
int time_to_int(string);
vector <EList> file_to_text(string);
void next_event(vector <EList>);
string free_time(vector <EList>);
void output_first(string);
void output_second(string);
string int_to_time(int);
string int_to_time(Time);

#endif
```

function.cpp

```cpp
#include "function.hpp"
#include <iostream>
#include <fstream>
#include <string>
#include <stdio.h>
#include <vector>

using namespace std;

string to_do_list(){
    string name;
    EList text;
    string task_name, time_start, duration = "0", preTime="";

//  вводимо ім'я файлу
    cout << "Enter input file: ";
    cin >> name;
    name = name + ".dat";

    ofstream fout(name, ios::binary);

    char new_line = 'y';
    while(new_line == 'y'){
        cout<<"Enter a name of task(maximum number of characters - 20): ";
        cin>>task_name;
        task_name = verifyName(task_name);//перевірка вводу
        text.name = task_name;

        cout<<"Enter an event start time in this format HH:MM: ";
        cin>>time_start;
        time_start = verifyTime(time_start, duration, preTime);//перевірка вводу
        preTime = time_start;
        text.time_start = time_start;


        cout<<"Enter a duration in this format HH:MM: ";
        cin>>duration;
        duration = verifyDuration(duration, time_start);//перевірка вводу
        text.duration = duration;


        fout.write((char*)&text, sizeof(EList));//запис у файл
```

```
        if(duration_normal(duration, time_start)){//перевірка, що залишився час для ще одної події
            cout << "\nDo you want to continue input task?[y/n]: ";
            cin >> new_line;
        }
        else{
            new_line = 'n';
        }
        cout<<endl;
    }


    fout.close();

    return name;
}


string verifyName(string name){
    //перевірка слова за кількістю літер
    while (name.size()>20 || name.size()<1){
        cout << "Enter a name again: ";
        cin >> name;
    }
    return name;
}

string verifyTime(string time, string duration, string preTime){
    bool flag_hours = hours(time);//перевірка правильність написання часу
    bool flag_duration = dur(time, duration, preTime);//перевірка на сумісність часу події з тривалістю минулої

    while(time.size()!=5 || time[2]!=':' || flag_hours || flag_duration){
        cout << "Enter a time again: ";
        cin >> time;
        flag_hours = hours(time);
        flag_duration = dur(time, duration, preTime);
    }
    return time;
}

```

```
string verifyDuration(string duration, string time){
    bool flag_hours = hours(duration);//перевірка правильність написання часу
    bool flag_dif = false;//перевірка, що подія не може закінчитися наступного для
    if(time_to_int(duration)+time_to_int(time) > 1440) flag_dif = true;
    while(duration.size()!=5 || duration[2]!=':' || flag_hours || flag_dif){
        cout << "Enter a duration again: ";
        cin >> duration;
        flag_hours = hours(duration);
        flag_dif = false;
        if(time_to_int(duration)+time_to_int(time) > 1440) flag_dif = true;
    }
    return duration;
}

bool hours(string time){
    bool flag = true;
    string h, m;
    h = time.substr(0, 2);
    m = time.substr(3, 2);
    int number_h = atoi(h.c_str()), number_m = atoi(m.c_str());

    if(number_h>=0 && number_h<=23){
        if(number_m>=0 && number_m<=59){
            flag = false;
        }
    }
    return flag;
}

bool dur(string time, string duration, string preTime){
    bool flag = false;
    int number = time_to_int(time);//переводимо час у кількість хвилин
    if(preTime!=""){
        int time_before = time_to_int(preTime);
        int d = time_to_int(duration);
        if(time_before + d > number){
            flag = true;
        }
    }
    return flag;
}
```

```
int time_to_int(string time){
    string h, m;
    h = time.substr(0, 2);
    m = time.substr(3, 2);
    int number = atoi(h.c_str())*60 + atoi(m.c_str());
    return number;
}

bool duration_normal(string duration, string time){
    bool flag = true;
    if(time_to_int(duration) + time_to_int(time) == 1440) flag = false;
    return flag;
}

void output_first(string name){//виводимо текст файлу
    EList text;
    ifstream fin(name, ios::binary);//відкриваємо файл для читання

    cout<<"File "<< name <<": \n";

    while (fin.read((char*)&text, sizeof(EList))){
        cout<< text.name <<", "<<  int_to_time(text.time_start)<<", "<<"duration: "<< int_to_time(text.duration)<< ";"<<endl;
    }
    fin.close();
    cout<<endl;
}

void output_second(string name){//виводимо текст файлу
    FreeList text;
    ifstream fin(name, ios::binary);//відкриваємо файл для читання

    cout<<"File "<< name <<": \n";

    while (fin.read((char*)&text, sizeof(FreeList))){
        cout<< int_to_time(text.start) <<"-"<< int_to_time(text.finish)<<", "<<"duration: "<< int_to_time(text.free_dur)<<
            ";"<<endl;
    }

    fin.close();
    cout<<endl;
}
```

```cpp
vector <EList> file_to_text(string name){
    vector<EList> pack;
    EList text;
    ifstream fin(name, ios::binary);//відкриваємо файл для читання

    while (fin.read((char*)&text, sizeof(EList))){
        pack.push_back(text);
    }
    fin.close();
    cout<<endl;

    return pack;
}

void next_event(vector <EList> text){
    int j;
    int hour_task, minute_task;
    time_t rawtime;
    int time_now, time_task;
    int dif, min = 1440;
    string m;
    struct tm * timeinfo;//знаходимо час цього моменту
    time( &rawtime );
    timeinfo = localtime ( &rawtime );
    int hour = timeinfo->tm_hour, minute = timeinfo->tm_min;
    //виводимо час у правильному форматі
    if(hour>=10 && minute<10) cout<<"Time now: "<< hour << ":" << "0"<< minute <<endl;
    if(hour<10 && minute<10) cout<<"Time now: "<< "0" << hour << ":" << "0"<< minute <<endl;
    if(hour<10 && minute>=10) cout<<"Time now: "<< "0" << hour << ":" << minute <<endl;
    if(hour>=10 && minute>=10) cout<<"Time now: "<< hour << ":" << minute <<endl;
    time_now = hour * 60 + minute;
    for(int i = 0; i < text.size(); i++){
        hour_task = text[i].time_start.hours;
        minute_task = text[i].time_start.minutes;
        time_task = hour_task * 60 + minute_task;
        dif = time_task - time_now;
        if(dif>0 && dif<min){
            j = i;
            min = dif;
        }
    }
```

Variable 'dif' may be uninitializ

Variable 'j' may be uninitializ

```
    //знаходимо наступну подію, якщо така існує
    if(dif>0){                                                    Variable 'dif' may be uninitialized when used he
        cout<<"Next event: "<< text[j].name <<", "<<  int_to_time(text[j].time_start)<<", "<<"duration: "<<
            int_to_time(text[j].duration)<< ";"<<endl<<endl;
    }
    else cout<<"Today you have no more business!!!"<<endl<<endl;
}

string free_time(vector <EList> text){
    FreeList file;
    string s, i;
    int time_n, dur_n;
    string start = "13:00", finish = "23:59", free_dur = "10:59";
    bool task_flag = false;
    string name2;
    cout<<"Enter output file: "; cin>>name2;//називаємо другий файл

    name2+=".dat";
    ofstream fout(name2, ios::binary);

    for(int i = 0; i < text.size(); i++){
        time_n = text[i].time_start.hours * 60 + text[i].time_start.minutes;
        dur_n = text[i].duration.hours * 60 + text[i].duration.minutes;
        if(time_n + dur_n>=780){//події після 13:00
            start = int_to_time(time_n + dur_n);//кількість хвилин переводимо у правильний формат часу
            if(i + 1 < text.size()){
                finish = int_to_time(text[i+1].time_start.hours * 60 + text[i+1].time_start.minutes);
            }
            else finish = "23:59";
            free_dur = int_to_time(time_to_int(finish) - time_to_int(start));
            file.start = start;
            file.finish = finish;
            file.free_dur = free_dur;
            fout.write((char*)&file, sizeof(FreeList));//записуємо у файл
            task_flag = true;//флаг, що у другій половині дня є події
        }
    }
    if(!task_flag){//фільний час, якщо після 13:00 подій нема
        file.start = start;
        file.finish = finish;
        file.free_dur = free_dur;
        fout.write((char*)&file, sizeof(FreeList));
    }

    fout.close();

    return name2;
```

```
string int_to_time(int num){
    string time;
    int h, m;
    h = num/60;
    m = num%60;
    if(h<10 && m<10) time ="0" + to_string(h) + ":" + "0" + to_string(m);
    if(m<10 && h>=10) time = to_string(h) + ":" + "0" + to_string(m);
    if(h<10 && m>=10) time = "0" + to_string(h) + ":" + to_string(m);
    if(h>=10 && m>=10) time = to_string(h) + ":" + to_string(m);
    return time;
}

string int_to_time(Time num){
    string time;
    int h, m;
    h = num.hours;
    m = num.minutes;
    if(h<10 && m<10) time ="0" + to_string(h) + ":" + "0" + to_string(m);
    if(m<10 && h>=10) time = to_string(h) + ":" + "0" + to_string(m);
    if(h<10 && m>=10) time = "0" + to_string(h) + ":" + to_string(m);
    if(h>=10 && m>=10) time = to_string(h) + ":" + to_string(m);
    return time;
}

```

Код Python

Lab2.py

```python
from function import *

# називаємо вихідний файл
print("Enter input file: ")
name = input()
name = name + ".dat"

# вводимо текст з консолі
size = capture_text(name)

# записуємо текст з файлу name у змінну text
text = read_file(name, size)

#знаходимо наступну подію
next_event(text)

# називаємо створений файл
name2 = "New" + name
# переписуємо компоненти тексту
new_size = free_time(text, name2)
#читаємо файл
read_file2(name2, new_size)

```

function.py

```python
from datetime import datetime
import pickle

#клас для часу
class Time:

    minutes = 0
    hours = 0

    def createTime(args):
        time = Time()
        time.hours = int(args[:2])
        time.minutes = int(args[3:])

        return time

    def toString(self):

        if self.hours < 10 and self.minutes < 10:
            time = "0" + str(self.hours) + ":" + "0" + str(self.minutes)
        elif self.minutes < 10 and self.hours >= 10:
            time = str(self.hours) + ":" + "0" + str(self.minutes)
        elif self.hours < 10 and self.minutes >= 10:
            time = "0" + str(self.hours) + ":" + str(self.minutes)
        elif self.hours >= 10 and self.minutes >= 10:
            time = str(self.hours) + ":" + str(self.minutes)
        return time

    def __init__(self):
        self.minutes = int()
        self.hours = int()

```

```python
#клас для подій
class Task:
    name = str()
    time_start = Time()
    duration = Time()

    def __init__(self,name,time_start,duration):
        self.name = name
        self.time_start = Time.createTime(time_start)
        self.duration = Time.createTime(duration)

#клас для вільного часу
class FreeT:
    start = Time()
    finish = Time()
    free_dur = Time()

    def __init__(self,start,finish,free_dur):
        self.start = Time.createTime(start)
        self.finish = Time.createTime(finish)
        self.free_dur = Time.createTime(free_dur)



def time_to_int(time: str):#час у кількість хвилин
    h = time[:2]
    m = time[3:]
    number = int(h) * 60 + int(m)
    return number

```

```
63  def hours(time: str):
64      flag = True
65      h = time[:2]
66      m = time[3:]
67      if int(h) >= 0 and int(h) <= 23:
68          if int(m) >= 0 and int(m) <= 59:
69              flag = False
70
71      return flag
72
73  def dur(time: str, duration: str, times: str):
74      flag = False
75      number = time_to_int(time)
76      if len(time) != 0:
77          if len(times) != 0:
78              num_t = len(times)-1
79              time_before = time_to_int(times[num_t])
80              d = time_to_int(duration)
81              if time_before + d > number:
82                  flag = True
83      return flag
84
85  def verifyName(name: str):#перевіряємо кількість символів
86      while (len(name) > 20 or len(name) < 1):
87          print("Enter a name again: ")
88          name = input();
89      return name
90
```

```python
def verifyTime(time: str, duration, times):
    flag_hours = hours(time)#перевіряємо правильний формат часу
    flag_duration = dur(time, duration, times)#перевіряємо, що сумісність події та тривалості минулої
    while not (len(time) == 5 and time[2] == ':' and not flag_hours and not flag_duration):
        print("Enter a time again: ")
        time = input()
        flag_hours = hours(time)
        flag_duration = dur(time, duration, times)

    return time

def verifyDuration(duration: str, time: str):
    flag_hours = hours(duration)#перевіряємо правильний формат часу
    flag_dif = False#перевіряємо, що подія не може тривати до наступного дня
    if time_to_int(duration) + time_to_int(time) > 1440:
        flag_dif = True
    while len(duration) != 5 or duration[2] != ':' or flag_hours or flag_dif:
        print('Enter a duration again: ')
        duration = input()
        flag_hours = hours(duration)
        flag_dif = False
        if time_to_int(duration) + time_to_int(time) > 1440:
            flag_dif = True
    return duration

def duration_normal(duration: str, time: str):
    flag = True
    if time_to_int(duration) + time_to_int(time) == 1440:
        flag = False
    return flag

```

```python
def int_to_time(num: int):#кількість хвилин у правильний формат часу
    h = num // 60
    m = num % 60
    if h < 10 and m < 10:
        time ="0" + str(h) + ":" + "0" + str(m);
    elif m < 10 and h >= 10:
        time = str(h) + ":" + "0" + str(m);
    elif h < 10 and m >= 10:
        time = "0" + str(h) + ":" + str(m);
    elif h >= 10 and m >= 10:
        time = str(h) + ":" + str(m);
    return time;


def capture_text(name_f):#читаємо з консолі
    """
    Читаємо текст строками
    """
    flag = 'y'
    duration = '0'
    times = []
    size = 0

    with open(name_f, 'wb') as file:
        while flag == 'y':
            print('Enter a name of task(maximum number of characters - 20): ')
            task_name = input()
            task_name = verifyName(task_name)#перевіряємо ввод


            print('Enter a event start time in this format HH:MM: ')
            time_start = input()
            time_start = verifyTime(time_start, duration, times)#перевіряємо ввод
            times.append(time_start)

```

```
            print('Enter a duration in this format HH:MM: ')
            duration = input()
            duration = verifyDuration(duration, time_start)#перевіряємо ввод


            tasks = Task(task_name,time_start,duration)#записуємо у структуру

            pickle.dump(tasks, file)#записуємо у файл

            size+=1#розмір масиву подій

            if duration_normal(duration, time_start):#перевіряємо, що є час для наступної події
                print('\nDo you want to continue input task?[y/n]: ')
                flag = input()
            else:
                flag = 'n'


    return size

def write_text(name_f: str, text):
    """
    Записуємо текст у файл
    """
    with open(name_f, "wb") as file:#відкриваємо файл для запису
        for i in range(len(text)):
            pickle.dump(text[i], file)

```

```
185 def next_event(text):
186     min = 1440
187     current_datetime = datetime.now()#час зараз
188     h = current_datetime.hour
189     m = current_datetime.minute
190     if h >= 10 and m < 10:#правильний формат
191         print('Time now: ' + str(h) + ':' + '0' + str(m))
192     elif h < 10 and m < 10:
193         print('Time now: ' + '0' + str(h) + ':' + '0' + str(m))
194     elif h < 10 and m >= 10:
195         print('Time now: ' + '0' + str(h) + ':' + str(m))
196     elif h >= 10 and m >= 10:
197         print('Time now: ' + str(h) + ':' + str(m))
198     dif = 0
199     j = 0
200     time_now = h * 60 + m
201     for i in range(len(text)):#час подій
202         ht = text[i].time_start.hours
203         mt = text[i].time_start.minutes
204         time_list = ht * 60 + mt
205         dif = time_list - time_now
206         if dif > 0 and dif < min:
207             j = i
208             min = dif
```

```
    if dif > 0:#знаходимо наступну подію, якщо така є
        print('Next event: ' + text[j].name + ", " + text[j].time_start.toString() + ", " + "duration: " + text[j].duration.toString())
    else:
        print('Today you have no more business!!!')

def free_time(text, name_f):
    size = 0
    task_flag = False
    start = "13:00"
    finish = "23:59"
    free_dur = "10:59"
    file = open(name_f, "wb")
    for i in range(len(text)):
        time_n = text[i].time_start.hours * 60 + text[i].time_start.minutes
        dur_n = text[i].duration.hours * 60 + text[i].duration.minutes
        if time_n + dur_n >= 780:#події, які тривають після 13:00
            start = int_to_time(time_n + dur_n)
            if i + 1 < len(text):
                finish = int_to_time(text[i+1].time_start.hours * 60 + text[i+1].time_start.minutes)
            else:
                finish = '23:59'
            free_dur = int_to_time(time_to_int(finish) - time_to_int(start))
            free_text = FreeT(start, finish, free_dur)
            pickle.dump(free_text, file)
            task_flag = True#флаг, що подіє є після 13:00
            size+=1#розмір масиву
    if task_flag == False:#якщо подій нема, автоматичний вільний час
        free_text = FreeT(start, finish, free_dur)
        pickle.dump(free_text, file)

    return size
```

```
241     def read_file(name_f: str, size):
242         """
243         Читає текст з файлу
244         """
245         text = []
246         print("Text on input:" + "\n")
247         print(size)
248         with open(name_f, "rb") as file:#зчитуємо файл
249             for i in range(size):
250                 text.append(pickle.load(file))
251                 str_t = text[i].name + ", " + text[i].time_start.toString() + ", " + "duration: " + text[i].duration.toString()
252                 print(str(str_t) + "\n")
253
254         return text
255
256     def read_file2(name_f: str, size):
257         """
258         Читає текст з файлу
259         """
260         text = []
261         print("Text on output:" + "\n")
262         with open(name_f, "rb") as file:#зчитуємо файл
263             for i in range(size):
264                 text.append(pickle.load(file))
265                 str_t = text[i].start.toString() + "-" + text[i].finish.toString() + ", " + "duration: " + text[i].free_dur.toString()
266                 print(str(str_t) + "\n")
267
```

Результат

C++

```
Enter input file: f1
Enter a name of task(maximum number of characters - 20): atsgavhfbhbashfbhabshbfhahs
Enter a name again: task1
Enter an event start time in this format HH:MM: 90:00
Enter a time again: 12:30
Enter a duration in this format HH:MM: 02:00

Do you want to continue input task?[y/n]: y

Enter a name of task(maximum number of characters - 20): task2
Enter an event start time in this format HH:MM: 17:00
Enter a duration in this format HH:MM: 01:00

Do you want to continue input task?[y/n]: n


File f1.dat:
task1, 12:30, duration: 02:00;
task2, 17:00, duration: 01:00;


Time now: 14:56
Next event: task2, 17:00, duration: 01:00;

Enter output file: f2
File f2.dat:
14:30-17:00, duration: 02:30;
18:00-23:59, duration: 05:59;


Program ended with exit code: 0
```

Python

```
Enter input file:
f1
Enter a name of task(maximum number of characters - 20):
atsvhahsbjfbajsbjfbjasbfjasbjf
Enter a name again:
task1
Enter a event start time in this format HH:MM:
90:84
Enter a time again:
12:30
Enter a duration in this format HH:MM:
23:59
Enter a duration again:
02:00

Do you want to continue input task?[y/n]:
y
Enter a name of task(maximum number of characters - 20):
17:00
Enter a event start time in this format HH:MM:
18:00
Enter a duration in this format HH:MM:
03:00

Do you want to continue input task?[y/n]:
n
Text on input:

2
task1, 12:30, duration: 02:00

17:00, 18:00, duration: 03:00
```

```
Time now: 14:59
Next event: 17:00, 18:00, duration: 03:00
Text on output:

14:30-18:00, duration: 03:30

21:00-23:59, duration: 02:59


Process finished with exit code 0
```