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»
«Бінарні файли»
Варіант 7

Виконав студент ІП-15, Гуменюк Олександр Володимирович
(шифр, прізвище, ім'я, по батькові)
Перевірила Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 2

Бінарні файли

Індивідуальне завдання

Варіант 7

7. Створити файл з розкладом руху приміських поїздів декількома напрямками (по кожному напрямку по 3-5 рейсів протягом дня): номер рейсу, напрямок руху, час відправлення, час прибуття в кінцевий пункт. На основі даного розкладу сформувати зимовий розклад (новий файл), в якому мають бути тільки ранкові (до 10:00) та вечірні (після 18:00) рейси.

Код C++

OP_Lab1.cpp

```cpp
#include "functions.h"


int main()
{
    int mode = chooseMode();                                        // Choosing file opening mode (overwriting/adding)
    vector <TrainTrip> trips = generateTrips();                     // Generate vector of train trips with routes based on users input

    string scheduleFileName = "schedule.dat";                       // File name with whole schedule
    string winterScheduleFileName = "winter_schedule.dat";          // File name with winter schedule

    createSchedule(scheduleFileName, trips, mode);                  // Creating file with schedule
    createWinterSchedule(scheduleFileName, winterScheduleFileName); // Creating file with winter schedule

    printSchedule(scheduleFileName, "Schedule");                    // Outputting file with schedule in console
    printSchedule(winterScheduleFileName, "Winter Schedule");       // Outputting file with winter schedule in console
}
```

functions.h

```cpp
#pragma once
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <math.h>
#include <vector>

using namespace std;
const size_t kRouteNameSize = 15;

struct TrainTime { ... };

struct TrainTrip { ... };

int chooseMode();
vector <TrainTrip> generateTrips();
void createSchedule(string fileName, vector <TrainTrip> trips, int mode);
void createWinterSchedule(string scheduleFileName, string winterScheduleFileName);
void printSchedule(string fileName, string header);
```

functions.cpp

```cpp
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <math.h>
#include <vector>

using namespace std;
const size_t kRouteNameSize = 15;               //Constant size of char array - name of trip routes

/* Defines time that is used for train schedules.
 * Consists of number of hours and number of minutes
 */
struct TrainTime {
    int hours;
    int minutes;
};


/* Defines structure of train trips that are used for train schedules.
 * Consists of trip number (ID), trip route (direction),
 * and variables with custom-made time type: departure time and arrival time
 */
struct TrainTrip {
    int tripNum;
    char tripRoute[kRouteNameSize];
    TrainTime departureTime;
    TrainTime arrivalTime;
};
```

```cpp
/* Generates departure and arrival times with hours number based on "hours" and random minutes number.
 * Round number of hours and minutes, so they can't be greater than 23 and 59 respectively
 */
void generateTrainTime(int hours, TrainTime& depTime, TrainTime& arrTime) {
    depTime = { rand() % 24, rand() % 60 };
    arrTime = { (depTime.hours + hours) % 24, rand() % 60 };
}


/* Generates 3-5 trip with route name "routeName" and trip number "tripNumber" (with each trip tripNumber increases by one).
 * Each trip last for about "tripLength" hours. Appends generated trips to "trips" vector
 */
void generateRouteTrips(char routeName[kRouteNameSize], int& tripNumber, int tripLength, vector <TrainTrip>& trips) {

    for (int i = 0; i < rand()%3 + 3; i++) {
        TrainTrip trip;
        trip.tripNum = tripNumber;
        generateTrainTime(tripLength, trip.departureTime, trip.arrivalTime);
        strncpy_s(trip.tripRoute, routeName, kRouteNameSize);
        trips.push_back(trip);
        tripNumber++;
    }
}
```

```cpp
// This enum and function is used to let user choose the mode for information input (overwriting or adding)
enum Mode {
    OVERWRITING = 1,                                        // Overwriting mode: program erases any text that the file might have
    ADDING = 2                                              // Adding mode: program writes text on top of text from the file
};
int chooseMode() {
    string ch;                                              // Character that user writes to choose mode
    cout << "Choose writing mode (1 - for overwriting file, 2 - for adding to file): ";
    cin >> ch;

    while (ch != "1" && ch != "2") {                        // The function will ask for user input, until it's "1" or "2"
        cout << "You may only enter a '1' or an '2'!" << endl;
        cout << "Choose writing mode (1 - for overwriting file, 2 - for adding to file): ";
        cin >> ch;
    }

    return stoi(ch);                                        // The function returns an int 1, which in enum Mode stands for OVERWRITING,
}
```

```cpp
// Checks if string "s" constists only of digit characters
bool isNum(string s)
{
    for (char ch : s) {
        if (!isdigit(ch)) return false;
    }
    return true;
}


// Allows to input only positive integers
int inputNum(string text) {
    string n;
    cout << text << ": ";
    cin >> n;

    while (!isNum(n)) {
        cout << "You can only enter a positive integer: ";
        cin >> n;
    }
    cin.ignore();
    return stoi(n);
}
```

```cpp
/* Asks user for number of routes and name for each one.
 * Then generates train trips for each route, using function "generateRouteTrips"
 * Returns a vector with all train trips
 */
vector <TrainTrip> generateTrips() {
    srand(unsigned(time(NULL)));

    int numOfRoutes = inputNum("Enter the number of routes (directions)");      // Number of all routes

    int tripNumber = 11111;                                                     // Trip number starts 11111 and increases by one with each trip
    vector <TrainTrip> trips;                                                   // Vector, consisted of all trips

    for (int i = 0; i < numOfRoutes; i++) {
        char routeName[kRouteNameSize];
        cout << "Enter name of route #" << i + 1 << ": ";
        cin.getline(routeName, kRouteNameSize);
        generateRouteTrips(routeName, tripNumber, rand() % 6 + 2, trips);
    }

    return trips;
}
```

```cpp
// Writes all trips from vector "trips" into file with name "fileName" with writing mode "mode"
void createSchedule(string fileName, vector <TrainTrip> trips, int mode) {

    ofstream file;

    if (Mode::OVERWRITING == mode) {
        file.open(fileName, ios::binary);
    }
    else {
        file.open(fileName, ios::binary | ios::app);
    }

    for (TrainTrip trip : trips) {
        file.write(reinterpret_cast<char*>(&trip), sizeof(TrainTrip));
    }
    file.close();
}
```

```cpp
// Checks whether a trip is a winter trip. Winter trips are trips before 10:00 or after 18:00
bool isWinterTrip(TrainTrip trip) {

    bool isMorningDeparture = trip.departureTime.hours < 10;
    bool isEveningDeparture = trip.departureTime.hours > 18 || (trip.departureTime.hours == 18 && trip.departureTime.minutes != 0);

    bool isMorningArrival = trip.arrivalTime.hours < 10;
    bool isEveningArrival = trip.arrivalTime.hours > 18 || (trip.arrivalTime.hours == 18 && trip.arrivalTime.minutes != 0);

    /* A trip is winter only if it:
     * starts in the morning and finishes in the morning OR
     * starts in the evening and finishes in the evening OR
     * start in the evening and finishes in the morning
     */
    if ((isMorningDeparture && isMorningArrival) || (isEveningDeparture && (isEveningArrival || isMorningArrival))) {
        return true;
    }
    else {
        return false;
    }
}
```

```cpp
/* Reads all trips from file "scheduleFileName" and writes only winter one to vector "trips".
 * Using functions "createSchedule" writes all winter trips in the "winterScheduleFileName" file
 */
void createWinterSchedule(string scheduleFileName, string winterScheduleFileName) {

    vector <TrainTrip> trips;
    TrainTrip trip;
    ifstream schedule(scheduleFileName, ios::binary);

    while (schedule.read(reinterpret_cast<char*>(&trip), sizeof(TrainTrip))) {
        if (isWinterTrip(trip)) trips.push_back(trip);
    }

    schedule.close();

    createSchedule(winterScheduleFileName, trips, 1);
}
```

```cpp
// Prints all properties of "trip" in the console
void printTrip(TrainTrip trip) {
    cout << "Trip Number: " << trip.tripNum;
    cout << " Route: " << trip.tripRoute;
    cout << " Departure Time: " << setw(2) << setfill('0') << trip.departureTime.hours << ":" << setw(2) << setfill('0') << trip.departureTime.minutes;
    cout << " Arrival Time: " << setw(2) << setfill('0') << trip.arrivalTime.hours << ":" << setw(2) << setfill('0') << trip.arrivalTime.minutes << endl;
}


/* Writes string "header" first.
 * Reads all trips from "fileName" and prints them(their properties) in the console, using functions "printTrip".
 * Visually split output of trips with different routes into different sections.
 */
void printSchedule(string fileName, string header) {

    ifstream schedule(fileName, ios::binary);
    TrainTrip trip;
    char tripRoute[kRouteNameSize];

    cout << "\n" << header << "\n";

    if (schedule.read(reinterpret_cast<char*>(&trip), sizeof(TrainTrip))) {            // If file consists of at least one trip
        do {
            strncpy_s(tripRoute, trip.tripRoute, kRouteNameSize);                      // Copy the route name of trip to "tripRoute" (current route)
            cout << "Route Name: " << tripRoute << endl;
            printTrip(trip);

            while (schedule.read(reinterpret_cast<char*>(&trip), sizeof(TrainTrip)) && !strcmp(trip.tripRoute, tripRoute)) {
                printTrip(trip);                                                       // Prints all trips with route Name "routeName"
            }                                                                          // and break after first trips with different route name
            cout << "\n";                                                              // or after last trip in the file

        } while (!schedule.eof());                                                     // Prints trip until the end of the file
    }

    schedule.close();
}
```

Код Python

main.py

```python
import functions as f


def main():
    mode = f.chooseMode()                               # Choosing file opening mode (overwriting/adding)
    trips = f.generateTrips()                           # Generate list of train trips with routes based on users input

    scheduleFileName = "schedule.dat"                   # File name with whole schedule
    winterScheduleFileName = "winter_schedule.dat"      # File name with winter schedule

    f.createSchedule(scheduleFileName, trips, mode)                    # Creating file with schedule
    f.createWinterSchedule(scheduleFileName, winterScheduleFileName)    # Creating file with  winter schedule

    f.printSchedule(scheduleFileName, "Schedule")                      # Outputting file with schedule in console
    f.printSchedule(winterScheduleFileName, "Winter Schedule")         # Outputting file with winter schedule in console


if __name__ == '__main__':
    main()
```

structures.py

```python
# Defines class of time that is used for train schedules.
# Consists of number of hours and number of minutes
class TrainTime:
    def __init__(self, hours, minutes):
        self.hours = hours
        self.minutes = minutes


# Defines class of train trips that are used for train schedules.
# Consists of trip number (ID), trip route (direction),
# and variables with custom-made time type: departure time and arrival time
class TrainTrip:
    def __init__(self, tripRoute, tripNum, departureTime, arrivalTime):
        self.tripRoute = tripRoute
        self.tripNum = tripNum
        self.departureTime = departureTime
        self.arrivalTime = arrivalTime
```

functions.py

```python
import pickle
import random
import structures as struct


#Allows to input only positive integers
def inputNum(text):
    text += ": "
    num = input(text)
    while (not num.isdigit()) or int(num) < 0:
        print(f"{num} isn't a positive number!")
        num = input(text)
    return int(num)


# Generates departure and arrival times with hours number based on "hours" and random minutes number.
# Round number of hours and minutes, so they can't be greater than 23 and 59 respectively
def generateTrainTime(hours):
    depTime = struct.TrainTime(random.randint(0, 23), random.randint(0, 59))
    arrTime = struct.TrainTime((depTime.hours + hours) % 24, random.randint(0, 59))
    return depTime, arrTime


# Generates 3-5 trip with route name "routeName" and trip number "tripNumber" (with each trip tripNumber increases by
# one). Each trip last for about "tripLength" hours. Appends generated trips to "trips" list
def generateRouteTrips(tripRoute, tripNum, tripLength, trips):
    for i in range(random.randint(3, 5)):
        depTime, arrTime = generateTrainTime(tripLength)
        trip = struct.TrainTrip(tripRoute, tripNum, depTime, arrTime)
        trips.append(trip)
        tripNum += 1
    return tripNum
```

```python
# Asks user for number of routes and name for each one.
# Then generates train trips for each route, using function "generateRouteTrips"
# Returns a list with all train trips
def generateTrips():
    trips = []
    tripNum = 11111
    numOfRoutes = inputNum("Enter the number of routes (directions)")
    for i in range(numOfRoutes):
        tripRoute = input(f"Enter name of route #{i + 1} : ")
        tripNum = generateRouteTrips(tripRoute, tripNum, random.randint(2, 8), trips)
    return trips


# Lets user choose the mode for information input (overwriting or adding)
def chooseMode():
    ch = input("Choose writing mode (1 - for overwriting text, 2 - for adding text): ")
    while ch != "1" and ch != "2":
        print("You may only enter a '1' or an '2'!")
        ch = input("Choose writing mode (1 - for overwriting text, 2 - for adding text): ")

    if ch == "1":
        return "wb"
    else:
        return "ab"


# Writes all trips from list "trips" into file with name "fileName" with writing mode "mode"
def createSchedule(fileName, trips, mode):
    with open(fileName, mode) as file:
        for trip in trips:
            pickle.dump(trip, file)
```

```python
# Checks whether a trip is a winter trip. Winter trips are trips before 10:00 or after 18:00
def isWinterTrip(trip):
    isMorningDeparture = trip.departureTime.hours < 10
    isEveningDeparture = trip.departureTime.hours > 18 or (trip.departureTime.hours == 18 and trip.departureTime.minutes != 0)
    isMorningArrival = trip.arrivalTime.hours < 10
    isEveningArrival = trip.arrivalTime.hours > 18 or (trip.arrivalTime.hours == 18 and trip.arrivalTime.minutes != 0)

    if (isMorningDeparture and isMorningArrival) or (isEveningDeparture and (isEveningArrival or isMorningArrival)):
        return True
    else:
        return False


# Reads all trips from file "scheduleFileName" and writes only winter one to list "trips".
# Using functions "createSchedule" writes all winter trips in the "winterScheduleFileName" file
def createWinterSchedule(scheduleFileName, winterScheduleFileName):
    winterTrips = []
    with open(scheduleFileName, "rb") as schedule:
        while True:
            try:
                trip = pickle.load(schedule)
                if isWinterTrip(trip): winterTrips.append(trip)
            except EOFError:
                break

    createSchedule(winterScheduleFileName, winterTrips, "wb")
```

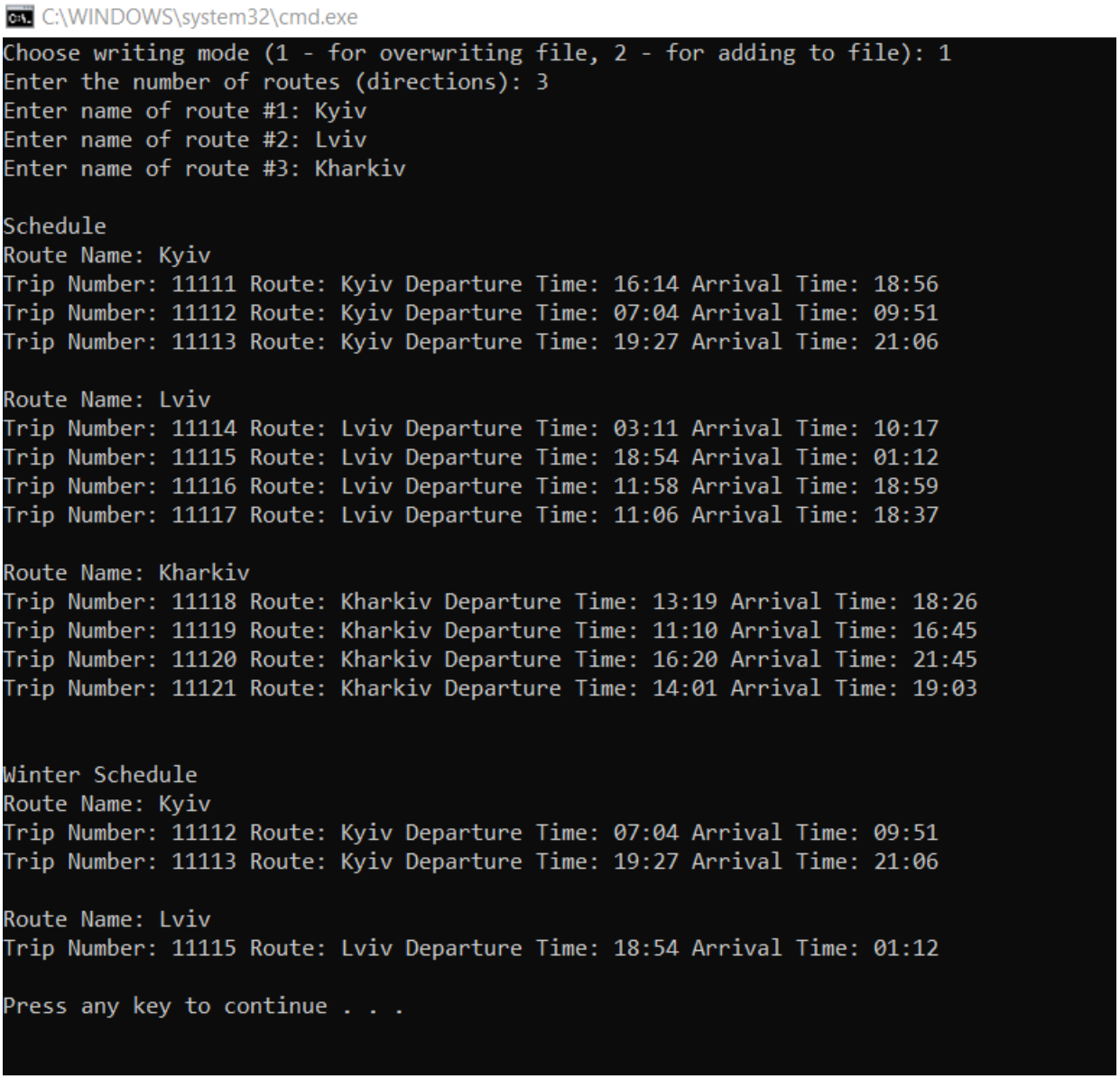
```python
#Prints all properties of "trip" in the console
def printTrip(trip):
    print(f"Trip Number: {trip.tripNum} Direction: {trip.tripRoute}", end=" ")
    print(f"Departure Time: {str(trip.departureTime.hours).zfill(2)}:{str(trip.departureTime.minutes).zfill(2)}",
          end=" ")
    print(f"Arrival Time: {str(trip.arrivalTime.hours).zfill(2)}:{str(trip.arrivalTime.minutes).zfill(2)}")


# Writes string "header" first.
# Reads all trips from "fileName" and prints them(their properties) in the console, using functions "printTrip".
# Visually split output of trips with different routes into different sections.
def printSchedule(fileName, header):
    print("\n" + header)

    trips = []
    with open(fileName, "rb") as schedule:
        while True:
            try:
                trips.append(pickle.load(schedule))
            except EOFError:
                break
    i = 0
    while i < len(trips):
        tripRoute = trips[i].tripRoute
        print(f"Direction Name: {tripRoute}")
        printTrip(trips[i])
        i += 1
        while i < len(trips) and tripRoute == trips[i].tripRoute:
            printTrip(trips[i])
            i += 1
        print("")
```

Тестування коду

C++



Python

```
"C:\Users\Productive Sasha\AppData\Local\Programs\Python\Python39\python.exe" "C:/Users/Productive Sasha/PycharmProjects/OP_Lab2/main.py"
Choose writing mode (1 - for overwriting text, 2 - for adding text): 1
Enter the number of routes (directions): 3
Enter name of route #1 : Lviv
Enter name of route #2 : Kyiv
Enter name of route #3 : Kharkiv

Schedule
Direction Name: Lviv
Trip Number: 11111 Direction: Lviv Departure Time: 09:32 Arrival Time: 12:21
Trip Number: 11112 Direction: Lviv Departure Time: 21:04 Arrival Time: 00:31
Trip Number: 11113 Direction: Lviv Departure Time: 03:19 Arrival Time: 06:04
Trip Number: 11114 Direction: Lviv Departure Time: 14:39 Arrival Time: 17:34

Direction Name: Kyiv
Trip Number: 11115 Direction: Kyiv Departure Time: 04:48 Arrival Time: 09:16
Trip Number: 11116 Direction: Kyiv Departure Time: 10:47 Arrival Time: 15:06
Trip Number: 11117 Direction: Kyiv Departure Time: 20:48 Arrival Time: 01:18
Trip Number: 11118 Direction: Kyiv Departure Time: 10:52 Arrival Time: 15:09

Direction Name: Kharkiv
Trip Number: 11119 Direction: Kharkiv Departure Time: 12:51 Arrival Time: 19:10
Trip Number: 11120 Direction: Kharkiv Departure Time: 00:56 Arrival Time: 07:21
Trip Number: 11121 Direction: Kharkiv Departure Time: 15:36 Arrival Time: 22:02
Trip Number: 11122 Direction: Kharkiv Departure Time: 01:20 Arrival Time: 08:15


Winter Schedule
Direction Name: Lviv
Trip Number: 11112 Direction: Lviv Departure Time: 21:04 Arrival Time: 00:31
Trip Number: 11113 Direction: Lviv Departure Time: 03:19 Arrival Time: 06:04

Direction Name: Kyiv
Trip Number: 11115 Direction: Kyiv Departure Time: 04:48 Arrival Time: 09:16
Trip Number: 11117 Direction: Kyiv Departure Time: 20:48 Arrival Time: 01:18

Direction Name: Kharkiv
Trip Number: 11120 Direction: Kharkiv Departure Time: 00:56 Arrival Time: 07:21
Trip Number: 11122 Direction: Kharkiv Departure Time: 01:20 Arrival Time: 08:15
```