# 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 № 9 з дисципліни

«Основи програмування

»

Варіант 23

Виконав студент Панченко Сергій Віталійович

(шифр, прізвище, ім'я, по батькові)

Перевірив

( прізвище, ім'я, по батькові)

Київ 202

## Лабораторна робота 2

**Мета:** вивчити особливості обробки текстових файлів даних та бінарних фалів

**Постановка задачі:**

**Бінарні файли:** розробимо клас клієнт, який характеризує ім’я клієнта, його прихід і вихід. Записуємо їх у файл і перевіряємо, чи не перетинається час обслуговування між двома клієнтами. Визначаємо клієнтів, яких обслуговували > 30 хв.

**Бінарні файли:**

C:

|  |
| --- |
| #include <iostream> |
|  | #include<fstream> |
|  | #include<vector> |
|  | #include<string> |
|  | #include<cmath> |
|  | using namespace std; |
|  | void create\_list(void); |
|  | void more\_half(void); |
|  | void show\_res(void); |
|  | struct cl { |
|  | char surname[255]; |
|  | int start\_serv; |
|  | int end\_serv; |
|  | }; |
|  | int main() |
|  | { |
|  | create\_list(); |
|  | more\_half(); |
|  | show\_res(); |
|  | system("pause"); |
|  | } |
|  | void show\_res() { |
|  | cl c; |
|  | ifstream file("clients30.txt", ios::binary); |
|  | while (file.read((char\*)&c, sizeof(cl))) { |
|  | cout << c.surname << "\t" << c.start\_serv << "\t" << c.end\_serv << endl; |
|  | } |
|  | file.close(); |
|  | } |
|  | void more\_half() { |
|  | ifstream file("clients.txt", ios::binary); |
|  | ofstream file2("clients30.txt", ios::binary); |
|  | vector<cl> cls; |
|  | cl c; |
|  | while (file.read((char\*)&c, sizeof(cl))) { |
|  | int h1 = int(c.start\_serv / 100); |
|  | int h2 = int(c.end\_serv / 100); |
|  | int m1 = int(c.start\_serv % 100); |
|  | int m2 = int(c.end\_serv % 100); |
|  | int h\_delay = abs(h2 - h1); |
|  | int m\_delay = abs(m2 - m1); |
|  | int delay = h\_delay\*60 + m\_delay; |
|  | if (delay > 30) { |
|  | cls.push\_back(c); |
|  | } |
|  | } |
|  | for (int i = 0; i < cls.size(); i++) |
|  | { |
|  | c = cls[i]; |
|  | file2.write((char\*)&c, sizeof(cl)); |
|  | } |
|  | file.close(); |
|  | file2.close(); |
|  | } |
|  | void create\_list() { |
|  | vector<cl> clients; |
|  | ofstream file("clients.txt", ios::binary); |
|  | string buf = "y"; |
|  | int count = 0; |
|  | bool is\_ok = true; |
|  | while (buf == "y") { |
|  | is\_ok = true; |
|  | count++; |
|  | cl c; |
|  | cout << "Client " << count << endl; |
|  | cout << "Surname: "; |
|  | cin >> c.surname; |
|  | cin.ignore(); |
|  | cout << "Service begin: "; |
|  | cin >> c.start\_serv; |
|  | cin.ignore(); |
|  | cout << "Service end: "; |
|  | cin >> c.end\_serv; |
|  | cin.ignore(); |
|  |  |
|  | for (auto i = clients.cbegin(); i != clients.cend()&&is\_ok; i++) { |
|  | if ((c.start\_serv >= i->start\_serv && c.start\_serv <= i->end\_serv) |
|  | ||(c.end\_serv >= i->start\_serv && c.end\_serv <= i->end\_serv)) { |
|  | is\_ok = false; |
|  | cout << "Mismatch in time" << endl;; |
|  | } |
|  | } |
|  | if (is\_ok) { |
|  | file.write((char\*)&c, sizeof(cl)); |
|  | clients.push\_back(c); |
|  | } |
|  | cout << "Add another client? y/n: "; |
|  | cin >> buf; |
|  | cin.ignore(); |
|  | } |
|  | file.close(); |
|  | } |

Python:

|  |
| --- |
| import pickle |
|  | class cl:  def \_\_init\_\_(self, surname, start\_serv, end\_serv):  self.surname = surname  self.start\_serv = start\_serv  self.end\_serv = end\_serv  def show\_res():  with open('clients30.txt', 'rb') as file:  bin\_data = file.read()  clients = pickle.loads(bin\_data)['clients']  for client in clients:  print(f'{client.surname}\t{client.start\_serv}\t{client.end\_serv}')  def more\_half():  with open('clients.txt', 'rb') as file:  cls = []  bin\_data = file.read()  clients = pickle.loads(bin\_data)['clients']  for c in clients:  h1 = c.start\_serv // 100  h2 = c.end\_serv // 100  m1 = c.start\_serv % 100  m2 = c.end\_serv % 100  h\_delay = abs(h2 - h1)  m\_delay = abs(m2 - m1)  delay = h\_delay \* 60 + m\_delay  if delay > 30:  cls.append(c)  with open('clients30.txt', 'wb') as file2:  pickle.dump({'clients': cls}, file2)  def create\_list():  clients = list()  with open('clients.txt', 'wb') as file:  buf = 'y'  count = 1  while buf == 'y':  is\_ok = True  print(f'Client {count}')  print('Surname: ', end='')  surname = input()  print('Start: ', end='')  start\_serv = int(input())  print('End: ', end='')  end\_serv = int(input())  c = cl(surname, start\_serv, end\_serv)  for client in clients:  if client.start\_serv <= c.start\_serv <= client.end\_serv \  or client.start\_serv <= c.end\_serv <= client.end\_serv:  is\_ok = False  print('Mismatch in time')  if is\_ok:  clients.append(c)  print('Add another client? y/n')  buf = input()  count += 1  pickle.dump({'clients': clients}, file) |
|  |  |
|  | def main(): |
|  | create\_list() |
|  | more\_half() |
|  | show\_res() |
|  |  |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | main() |

**Результат:**

Можна подивитися txt файли у папках в самому репозиторії

**Висновок:** під час лабораторної роботи навчилися працювати з бінарними файлами. Оскільки програми створюють файли та доповнюють їх, виконуючи умови поставленої до варіанту задачі, то лабораторна виконана правильно.