# 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 1

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

Текстові файли:

C:

|  |
| --- |
| #include <string.h> |
|  | #include <limits.h> |
|  | #include <direct.h> |
|  | #include<iostream> |
|  | #include <fstream> |
|  | using namespace std; |
|  | void create\_files(); |
|  | void fill\_file(char\* file\_name); |
|  | void fill\_common(); |
|  | int count\_rows(); |
|  | void show\_contents(char\* file\_name); |
|  | int main() |
|  | { |
|  | char file\_name1[\_MAX\_PATH] = "first.txt"; |
|  | char file\_name2[\_MAX\_PATH] = "second.txt"; |
|  | char file\_name3[\_MAX\_PATH] = "third.txt"; |
|  | create\_files(); |
|  | cout << "Enter strings for the first file" << endl; |
|  | fill\_file(file\_name1); |
|  | cout << "Enter strings for the second file" << endl; |
|  | fill\_file(file\_name2); |
|  | fill\_common(); |
|  | cout << "Common rows: " << count\_rows() << endl; |
|  | cout << "First file: " << endl; |
|  | show\_contents(file\_name1); |
|  | cout << "Second file: " << endl; |
|  | show\_contents(file\_name2); |
|  | cout << "Third file: " << endl; |
|  | show\_contents(file\_name3); |
|  | system("pause"); |
|  | return 0; |
|  | } |
|  | void show\_contents(char\* file\_name) { |
|  | ifstream file(file\_name, ios::in); |
|  | char buf[\_MAX\_PATH]; |
|  | while (!file.eof()) { |
|  | file.getline(buf, \_MAX\_PATH); |
|  | strcat\_s(buf, "\n"); |
|  | cout << buf; |
|  | } |
|  | file.close(); |
|  | } |
|  | int count\_rows() { |
|  | ifstream file("third.txt", ios::in); |
|  | char buf[\_MAX\_PATH]; |
|  | int count = 0; |
|  | while (!file.eof()){ |
|  | file.getline(buf, \_MAX\_PATH); |
|  | count++; |
|  | } |
|  | file.close(); |
|  | if (count) |
|  | count--; |
|  | return count; |
|  | } |
|  | void fill\_common(){ |
|  | ifstream file1("first.txt", ios::in); |
|  | ifstream file2("second.txt", ios::in); |
|  | ofstream file3("third.txt", ios::in); |
|  | char buf1[\_MAX\_PATH]; |
|  | char buf2[\_MAX\_PATH]; |
|  | while (!file1.eof()) { |
|  | file1.getline(buf1, \_MAX\_PATH); |
|  | while (!file2.eof()&strcmp(buf1,"")) { |
|  | file2.getline(buf2, \_MAX\_PATH); |
|  | if (!strcmp(buf1, buf2)) { |
|  | strcat\_s(buf2, "\n"); |
|  | file3 << buf2; |
|  | } |
|  | } |
|  | file2.clear(); |
|  | file2.seekg(0, ios::beg); |
|  | } |
|  | file1.close(); |
|  | file2.close(); |
|  | file3.close(); |
|  | } |
|  | void fill\_file(char\* file\_name) { |
|  | ofstream file(file\_name, ios::out | ios::trunc); |
|  | char buf[\_MAX\_PATH]; |
|  | cin.getline(buf, \_MAX\_PATH); |
|  | string str = buf; |
|  | while (str[0]!=5) { |
|  | str += "\n"; |
|  | file << str.c\_str(); |
|  | cin.getline(buf, \_MAX\_PATH); |
|  | str = buf; |
|  | } |
|  | file.close(); |
|  | } |
|  | void create\_files() { |
|  | ofstream file1("first.txt", ios::out|ios::trunc); |
|  | ofstream file2("second.txt", ios::out|ios::trunc); |
|  | ofstream file3("third.txt", ios::out | ios::trunc); |
|  | file1.close(); |
|  | file2.close(); |
|  | file3.close(); |
|  | } |

Python:

|  |
| --- |
| def create\_file(name): |
|  | with open(name, 'w') as file: |
|  | s = input() |
|  | while ord(s[0]) != 5: |
|  | file.write(s + '\n') |
|  | s = input() |
|  |  |
|  |  |
|  | def fill\_common(name1, name2, name3): |
|  | with open(name1, 'r') as f1: |
|  | with open(name3, 'w') as f3: |
|  | for l1 in f1: |
|  | if l1 != '': |
|  | with open(name2, 'r') as f2: |
|  | for l2 in f2: |
|  | if l1 == l2: |
|  | f3.write(l1) |
|  |  |
|  |  |
|  | def count\_rows(name): |
|  | count = 0 |
|  | with open(name, 'r') as file: |
|  | for \_ in file: |
|  | count += 1 |
|  | return count |
|  |  |
|  |  |
|  | def show\_file(name): |
|  | with open(name, 'r') as file: |
|  | for line in file: |
|  | print(line, end='') |
|  |  |
|  |  |
|  | def main(): |
|  | file\_name1 = 'first.txt' |
|  | file\_name2 = 'second.txt' |
|  | file\_name3 = 'third.txt' |
|  | print('Enter lines for the first file') |
|  | create\_file(file\_name1) |
|  | print('Enter lines for the second file') |
|  | create\_file(file\_name2) |
|  | fill\_common(file\_name1, file\_name2, file\_name3) |
|  | print(f'Total rows in result: {count\_rows(file\_name3)}') |
|  | print('Firts file: ') |
|  | show\_file(file\_name1) |
|  | print('Second file: ') |
|  | show\_file(file\_name2) |
|  | print('Third file: ') |
|  | show\_file(file\_name3) |
|  |  |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | main() |

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() |

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