**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет   
імені Петра Могили**

Факультет комп’ютерних наук

Кафедра «Інтелектуальних інформаційних систем»

**ЗВІТ**

*з лабораторної роботи № 5*

**Варіант № 8**

Дисципліна "Структури та організація даних"

Напрям підготовки: 122 Комп’ютерні науки

ЛР.ПЗ.01-201.1910109

***Cтудент*** *\_\_\_\_\_\_\_\_\_\_\_\_\_Грабовськиий Є.О.*

*(підпис)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_*

*(дата)*

***Викладач***

*\_\_\_\_\_\_\_\_\_\_\_\_Кірей К.О.\_\_\_ (підпис)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

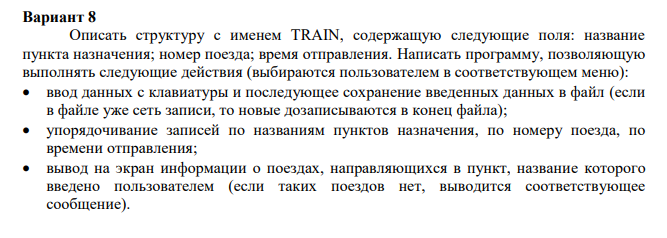
Миколаїв – 2020

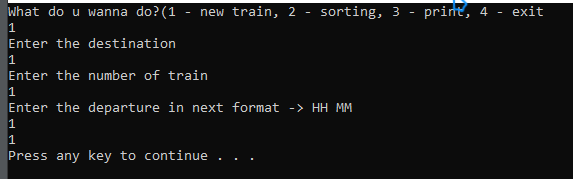
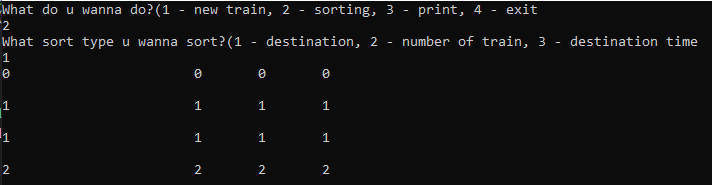
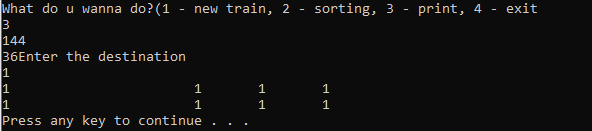
**ЗВІТ**

**Завдання**

Опис роботи

1. Открытые файла, перенос флага в конец – деление количества символов на размер структуры для получения количества экземпляров. Считывание файла в массив и сорировка.



1. Открытые файла**,** чтение в массив и выбор функции, которую хочет пользователь.**   Лістинг коду**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <cstdio>

#include <conio.h>

#include <locale>

#include <string>

#include <windows.h>

const int lenghtDestination = 20;

const int nameLength = 35;

const int yearLength = 5;

const int paymentLength = 10;

const int dbSize = 100;

const int lineLenght = nameLength + yearLength + paymentLength;

struct Employer

{

char name[nameLength + 1]{};

int year;

float salary;

void show() const

{

std::cout << name << '\t' << year << '\t' << salary << '\n';

}

};

struct Train

{

char destination[lenghtDestination + 1]{};

int number;

int departureH;

int departureM;

Train() : destination{}, number(0), departureH(0), departureM(0)

{

}

Train(const char destination[], int number, int departureH, int departureM)

{

int index = 0;

for (; destination[index] != '\0'; ++index)

{

this->destination[index] = destination[index];

}

strncpy(this->destination, destination, lenghtDestination);

this->number = number;

this->departureH = departureH;

this->departureM = departureM;

}

std::string toString()

{

std::string str;

for (int index = 0; index <= lenghtDestination;++index )

{

if(destination[index] == '\0')

str += ' ';

else

str += destination[index];

}

return str + '\t' + std::to\_string(number) +'\t' + std::to\_string(departureH) + '\t' + std::to\_string(departureM) + "\t\n";

}

};

class TrainHelper

{

public:

static int compareDestination(const void\* str1, const void\* str2)

{

return strcmp(((Train\*)str1)->destination, ((Train\*)str2)->destination);

}

static int compareNumber(const void\* str1, const void\* str2)

{

int leftValue = ((Train\*)str1)->number;

int rightValue = ((Train\*)str2)->number;

return leftValue > rightValue ? -1 : leftValue == rightValue ? 0 : 1;

}

static int compareTime(const void\* str1, const void\* str2)

{

int leftValue = ((Train\*)str1)->departureM + ((Train\*)str1)->departureH \* 60;

int rightValue = ((Train\*)str2)->departureM + ((Train\*)str2)->departureH \* 60;

return leftValue > rightValue ? -1 : leftValue == rightValue ? 0 : 1;

}

};

class EmployerHelper

{

public:

static int compareName(const void\* str1, const void\* str2)

{

return strcmp(((Employer\*)str1)->name, ((Employer\*)str2)->name);

}

static int compareSalary(const void\* str1, const void\* str2)

{

int leftValue = ((Employer\*)str1)->salary;

int rightValue = ((Employer\*)str2)->salary;

return leftValue > rightValue ? -1 : leftValue == rightValue ? 0 : 1;

}

static int compareYear(const void\* str1, const void\* str2)

{

int leftValue = ((Employer\*)str1)->year;

int rightValue = ((Employer\*)str2)->year;

return leftValue > rightValue ? -1 : leftValue == rightValue ? 0 : 1;

}

};

void clear();

void firstEx();

void secondEx();

void thirdEx();

void fourthEx();

void write(Train& train, FILE\* file);

int main()

{

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

secondEx();

//qsort(trains, 5, sizeof(train), compareNumber);

}

void firstEx()

{

FILE\* fbin;

//открытие файлф на чтение

fbin = fopen("db\_otd-kadr.bin", "rb");

if (fbin == NULL)

{

puts("Error while opening a file");

return;

}

//флаг переносится в конец

fseek(fbin, 0, SEEK\_END);

const int employeeCount = ftell(fbin) / sizeof(Employer);

Employer\* employee = new Employer[employeeCount];

Employer tmp;

std::cout << "Count of employee is " << employeeCount << '\n';

//флаг переносится в начало

fseek(fbin, 0, SEEK\_SET);

//чтение и запись в структуру

fread(employee, sizeof(Employer), employeeCount, fbin);

fclose(fbin);

int choice, choice2;

int j, i;

do {

puts("What sort type u wanna sort?(1 - salary, 2 - name, 3 - year of born, 4 - exit");

std::cin >> choice;

if (choice == 4)

{

break;

}

puts("Sort in reverse order?(1 - yes, 2 - no)");

//выбор сортировки

std::cin >> choice2;

switch (choice)

{

case 1:

qsort(employee, employeeCount, sizeof(Employer), EmployerHelper::compareSalary);

break;

case 2:

qsort(employee, employeeCount, sizeof(Employer), EmployerHelper::compareName);

break;

case 3:

qsort(employee, employeeCount, sizeof(Employer), EmployerHelper::compareYear);

break;

default:

break;

}

//действие при выборе обратной сортировки

if (choice2 == 1) {

i = employeeCount;

j = i - 1;

i = 0;

while (i < j)

{

tmp = employee[i];

employee[i] = employee[j];

employee[j] = tmp;

i++;

j--;

}

}

for (int index = 0; index < employeeCount; ++index)

{

printf("%s\t%i\t%f\n", employee[index].name, employee[index].year, employee[index].salary);;

}

system("pause");

clear();

} while (1);

}

void secondEx()

{

FILE\* fbin;

fbin = fopen("trains.bin", "rb");

if (fbin == NULL)

{

puts("Error while opening a file");

return;

}

Train train;

fseek(fbin, 0, SEEK\_END);

//расчёт размера файла

int employeeCount = ftell(fbin) / sizeof(train);

fseek(fbin, 0, SEEK\_SET);

fclose(fbin);

int choice, choice2;

int j, i;

do {

puts("What do u wanna do?(1 - new train, 2 - sorting, 3 - print, 4 - exit");

std::cin >> choice;

if (choice == 4)

{

break;

}

switch (choice)

{

case 1:

{

FILE\* file;

file = fopen("trains.bin", "a+b");

fseek(fbin, 0, SEEK\_END);

char destination[lenghtDestination];

int number, departureH, departureM;

char ch;

std::cout << "Enter the destination\n";

std::cin.ignore();

std::cin.getline(destination, sizeof(destination));

std::cout << "Enter the number of train\n";

std::cin >> number;

std::cout << "Enter the departure in next format -> HH MM\n";

std::cin >> departureH >> departureM;

//создание экземляра структуры для записи

train = Train(destination, number, departureH, departureM);

//запись структуры в файл

fwrite(&train, sizeof(train), 1, file);

fclose(file);

break;

}

case 2: {

std::cout << "What sort type u wanna sort?(1 - destination, 2 - number of train, 3 - destination time\n";

std::cin >> choice2;

FILE\* file;

file = fopen("trains.bin", "rb");

fseek(file, 0, SEEK\_END);

int employeeCount = ftell(file) / sizeof(train);

fseek(file, 0, SEEK\_SET);

Train\* trains = new Train[employeeCount];

//запись в структуру

fread(trains, sizeof(train), employeeCount, file);

//разработано класс TrainHelper для более удобного получение доступа к видам сортировки

switch (choice2)

{

case 1:

qsort(trains, employeeCount, sizeof(train), TrainHelper::compareDestination);

break;

case 2:

qsort(trains, employeeCount, sizeof(train), TrainHelper::compareNumber);

break;

case 3:

qsort(trains, employeeCount, sizeof(train), TrainHelper::compareTime);

break;

default:

break;

}

for (int index = 0; index < employeeCount; ++index)

std::cout << trains[index].toString() << '\n';

fclose(file);

break;

}

case 3: {

FILE\* file;

file = fopen("trains.bin", "rb");

fseek(file, 0, SEEK\_END);

int employeeCount = ftell(file) / sizeof(train);

std::cout << ftell(file) << '\n' << sizeof(train);

fseek(file, 0, SEEK\_SET);

Train\* trains = new Train[employeeCount];

fread(trains, sizeof(train), employeeCount, file);

fclose(file);

char ch;

std::cout << "Enter the destination\n";

char destination[lenghtDestination];

std::cin.ignore();

std::cin.getline(destination, sizeof(destination));

for (int index = 0; index <= employeeCount; ++index)

{

//сравнение строк на равенство и вывод в случае, если это та строка, которая ищется

if (strcmp(trains[index].destination, destination) == 0)

{

std::cout << trains[index].toString();

}

}

break;

}

default:

break;

}

system("pause");

clear();

} while (1);

}

void thirdEx()

{

std::FILE\* fin, \* fout;

fin = fopen("db\_otd-kadr.txt", "r");

fout = fopen("db\_otd-kadr.bin", "wb");

if (fin == NULL || fout == NULL)

{

puts("The file don`t exist\n");

return;

}

Employer employer = Employer();

while (!feof(fin))

{

//запись имени работника в структуру

fgets(employer.name, nameLength, fin);

//запись зп и года рождения в структуру

fscanf(fin, "%i%f\n", &employer.year, &employer.salary);

printf("%s\t%i\t%f\n", employer.name, employer.year, employer.salary);

//запись в файл данных о работнике

fwrite(&employer, sizeof(employer), 1, fout);

}

fclose(fin);

fclose(fout);

}

void clear() {

std::cout << "\x1B[2J\x1B[H";

}