**ЗВІТ**

**до лабораторної роботи № < 11.3 >**

**« Впорядкування та бінарний пошук у бінарному файлі »**

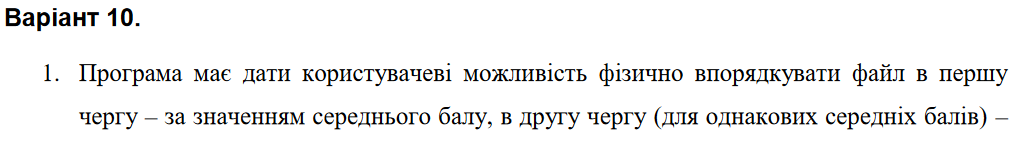
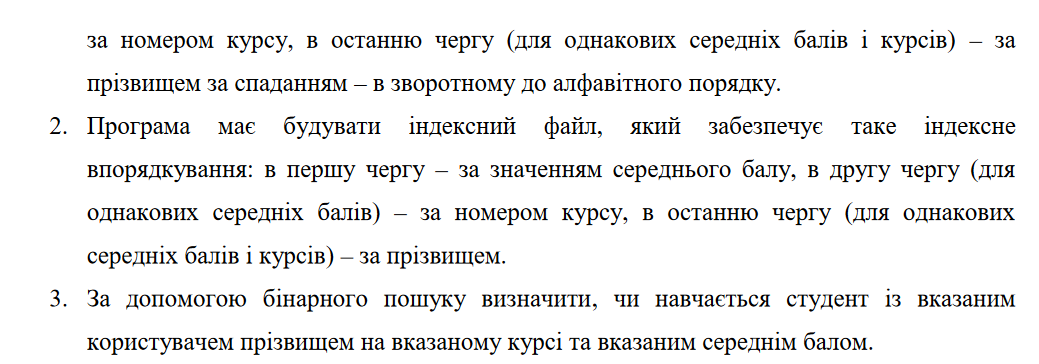
**з дисципліни**

**«Алгоритмізація та програмування»**

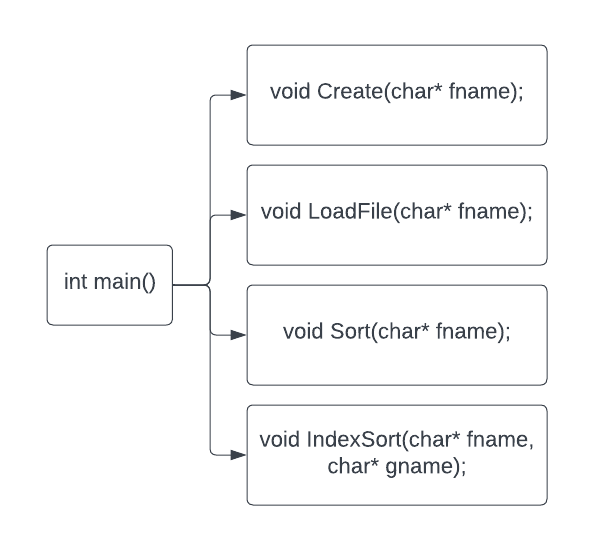
**студента групи ІН-105Б**

**Горанова Анастаса**

**Умова завдання:**

**Структурна схема:**



**Текст програми:**

// 11.3.cpp

// < Горанов Анастас >

// Лабораторна робота № 11.3

// Впорядкування та бінарний пошук у бінарному файлі

// Варіант 10

#include <iostream>

#include <stdio.h>

#include <iostream>

#include <fstream>

#include <errno.h>

#include <iomanip>

#include <string>

using namespace std;

enum Specialnist { KN, ME, IN, FI, TN };

string specialnistList[] = { "KN", "ME", "IN", "FI", "TN" };

struct Student

{

char surname[101];

int course;

Specialnist spec;

int fiz;

int mat;

double average;

union

{

int prog;

int chis\_metod;

int pedagog;

};

};

void Create(char\* fname);

void Print(char\* fname);

void Sort(char\* fname);

void IndexSort(char\* fname, char\* gname);

int main()

{

char fname[61];

char gname[61];

int menuItem;

do {

cout << endl;

cout << "Choose an action:" << endl << endl;

cout << " [1] - Entering data from the keyboard and saving" << endl;

cout << " [2] - loading and displaying data" << endl;

cout << " [3] - Sorting" << endl;

cout << " [4] - Index sorting and output" << endl;

cout << " [0] - Quit" << endl << endl;

cout << "Enter the value: "; cin >> menuItem;

cout << endl << endl << endl;

switch (menuItem)

{

case 1:

cin.get(); // очищуємо буфер клавіатури – щоб не було символу

cin.sync(); // "кінець рядка", який залишився після вводу числа

cout << "Enter a file name: "; cin.getline(fname, sizeof(fname));

Create(fname);

break;

case 2:

cin.get(); // очищуємо буфер клавіатури – щоб не було символу

cin.sync(); // "кінець рядка", який залишився після вводу числа

cout << "Enter a file name: "; cin.getline(fname, sizeof(fname));

Print(fname);

break;

case 3:

cin.get(); // очищуємо буфер клавіатури – щоб не було символу

cin.sync(); // "кінець рядка", який залишився після вводу числа

cout << "Enter a file name: "; cin.getline(fname, sizeof(fname));

Sort(fname);

cout << "Table was sorted!";

break;

case 4:

cin.get(); // очищуємо буфер клавіатури – щоб не було символу

cin.sync(); // "кінець рядка", який залишився після вводу числа

cout << "File name: "; cin.getline(fname, sizeof(fname));

cout << endl;

cout << "Output file name: "; cin.getline(gname, sizeof(gname));

cout << endl;

IndexSort(fname, gname);

cout << "Sorted file created!" << endl << endl;

Print(gname);

cout << endl;

case 0:

break;

default:

cout << "Invalid value" << endl;

}

} while (menuItem != 0);

return 0;

}

void Create(char\* fname)

{

ofstream f(fname, ios::binary);

if (!f)

{

cerr << "Error opening file '" << fname << "'" << endl;

f.close();

return;

}

int spec;

Student p;

char ch;

int i = 0;

do {

cout << "Student #" << i + 1 << ":" << endl;

cout << "Surname: "; cin >> p.surname;

cout << "Course: "; cin >> p.course;

cout << "Grades in physics: "; cin >> p.fiz;

cout << "Grades in math: "; cin >> p.mat;

cout << "A specialty (0 - KN, 1 - ME, 2 - IN, 3 - FI, 4 - TN): "; cin >> spec;

if (spec > 4) {

cout << "Wrong value, selected 2 (IN)" << endl;

spec = 0;

}

p.spec = (Specialnist)spec; // присвоєння спеціальності, введенної користувачем через перелік

switch (p.spec)

{

case KN:

cout << "Grades in programming: "; cin >> p.prog;

p.average = (p.mat + p.fiz + p.prog) / 3.0;

break;

case ME:

cout << "Grades in pedagogy: "; cin >> p.pedagog;

p.average = (p.mat + p.fiz + p.pedagog) / 3.0;

break;

case IN:

cout << "Grades in num.methods: "; cin >> p.chis\_metod;

p.average = (p.mat + p.fiz + p.chis\_metod) / 3.0;

break;

case FI:

cout << "Grades in pedagogy: "; cin >> p.pedagog;

p.average = (p.mat + p.fiz + p.pedagog) / 3.0;

break;

case TN:

cout << "Grades in pedagogy: "; cin >> p.pedagog;

p.average = (p.mat + p.fiz + p.pedagog) / 3.0;

break;

}

if (!f.write((char\*)&p, sizeof(Student)))

{

cerr << "Error writing to file." << endl;

f.close();

}

cout << "Continue? (Y/N) "; cin >> ch;

} while (ch == 'Y' || ch == 'y');

f.close();

}

void Print(char\* fname)

{

ifstream f(fname, ios::binary);

if (!f)

{

cerr << "Error opening file '" << fname << "'" << endl;

f.close();

return;

}

Student student;

cout << "==========================================================================================================="

<< endl;

cout << "| # | Surname | Course | Specilization | Physic | Mathematic | Programming | Numerical Methods | Pedagogy |"

<< endl;

cout << "-----------------------------------------------------------------------------------------------------------"

<< endl;

int i = 0;

while (f.read((char\*)&student, sizeof(Student)))

{

cout << "|" << setw(2) << right << i + 1 << " "; i++;

cout << "| " << setw(8) << left << student.surname

<< "| " << setw(4) << right << student.course << " "

<< " | " << setw(14) << left << specialnistList[student.spec]

<< setw(4) << "| " << setw(3) << left << student.fiz << " "

<< setw(6) << "| " << setw(6) << left << student.mat << " ";

switch (student.spec)

{

case KN:

cout << setw(8) << "| " << setw(5) << left << student.prog << setw(21)

<< " | " << setw(10) << left

<< "| " << left << " |" << endl;

break;

case ME:

cout << "" << setw(13) << "| " << " "

<< "" << setw(20) << "| "

<< setw(6) << "| " << setw(4) << left << student.pedagog << " |" << endl;

break;

case IN:

cout << "" << setw(13) << "| " << setw(10) << " |" << " "

<< setw(9) << student.chis\_metod << " |" << " "

<< setw(8) << "" << " |" << endl;

break;

case FI:

cout << "" << setw(13) << "| " << " "

<< "" << setw(20) << "| "

<< setw(6) << "| " << setw(4) << left << student.pedagog << " |" << endl;

break;

case TN:

cout << "" << setw(13) << "| " << " "

<< "" << setw(20) << "| "

<< setw(6) << "| " << setw(4) << left << student.pedagog << " |" << endl;

break;

}

}

cout << "==========================================================================================================="

<< endl;

cout << endl;

f.close();

}

void Sort(char\* fname)

{

ifstream f(fname, ios::binary);

if (!f)

{

cerr << "Error opening file '" << fname << "'" << endl;

f.close();

return;

}

f.seekg(0, ios::end); // курсор ставиться в кінець

int size = f.tellg(); // визначає величину файлу

size = size / sizeof(Student); // визначається кількість студентів

f.seekg(0, ios::beg);

Student\* student = new Student[size];

Student tmp;

f.read((char\*)student, size \* sizeof(Student));

f.close();

for (int i = 0; i < size - 1; i++) {

for (int j = size - 1; j > i; j--) {

string prizv1 = student[j].surname;

string prizv2 = student[j - 1].surname;

if ((student[j].average > student[j - 1].average)

||

(student[j].average == student[j - 1].average &&

student[j].course > student[j - 1].course)

||

(student[j].average == student[j - 1].average &&

student[j].course == student[j - 1].course &&

prizv1 > prizv2))

{

tmp = student[j];

student[j] = student[j - 1];

student[j - 1] = tmp;

}

}

}

f.close();

ofstream file(fname, ios::binary);

file.write((char\*)student, size \* sizeof(Student));

file.close();

}

void IndexSort(char\* fname, char\* gname)

{

ifstream f(fname, ios::binary);

if (!f)

{

cerr << "Error opening file '" << fname << "'" << endl;

f.close();

return;

}

f.seekg(0, ios::end);

int size = f.tellg();

size = size / sizeof(Student);

f.seekg(0, ios::beg);

Student\* student = new Student[size];

Student tmp;

f.read((char\*)student, size \* sizeof(Student));

for (int i = 0; i < size - 1; i++) {

for (int j = size - 1; j > i; j--) {

string prizv1 = student[j].surname;

string prizv2 = student[j - 1].surname;

if ((student[j].average > student[j - 1].average)

||

(student[j].average == student[j - 1].average &&

student[j].course > student[j - 1].course)

||

(student[j].average == student[j - 1].average &&

student[j].course == student[j - 1].course &&

prizv1 > prizv2))

{

tmp = student[j];

student[j] = student[j - 1];

student[j - 1] = tmp;

}

}

}

ofstream file(gname, ios::binary);

if (!file)

{

cerr << "Error opening file '" << gname << "'" << endl;

f.close();

return;

}

file.write((char\*)student, size \* sizeof(Student));

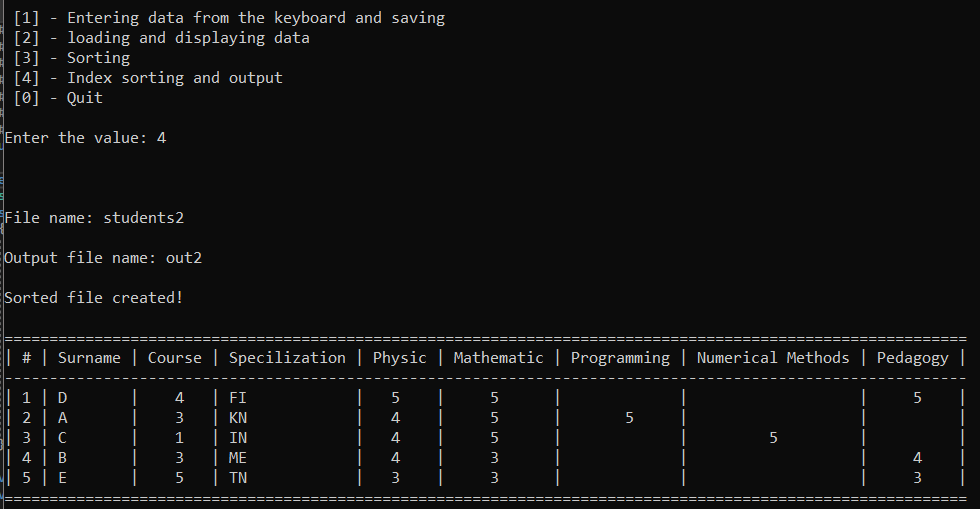
file.close();

}

**Посилання на git-репозиторій з проектом:**

*https://github.com/StassNG/11.3*

**Результати програми та unit-тесту:**

****

**Висновок:** я навчився опрацьовувати бінарні файли.