ЗВІТ

Про виконання лабораторної роботи №10.2B

*«Впорядкування та бінарний пошук в масиві структур»*

з дисципліни

«Алгоритмізація та програмування»

студента групи РІ-12

*Синчук Іван Романович*

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований описЗображення, що містить текст, знімок екрана, Шрифт, документ

Автоматично згенерований опис



#include <iostream>

#include <iomanip>

#include <string>

#include <algorithm>

#include <Windows.h>

using namespace std;

enum Specialization { COMPUTER\_SCIENCE, INFORMATICS, MATH\_ECONOMICS, PHYSICS\_INFORMATICS, LABOR\_EDUCATION };

string specializationStr[] = { "Комп'ютерні науки", "Інформатика", "Математика та економіка", "Фізика та інформатика", "Трудове навчання" };

struct Student {

string surname;

unsigned course;

Specialization specialization;

int physics;

int mathematics;

union {

int programming;

int numerical\_methods;

int pedagogy;

};

};

void InputStudents(Student\* students, int N);

void PrintStudents(const Student\* students, int N);

int GetProfileGrade(const Student& student);

void PhysicalSort(Student\* students, int N);

void IndexSort(const Student\* students, int N, int\* indexArray);

int BinarySearch(const Student\* students, const int\* indexArray, int N, const string& surname, int course, int profileGrade);

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int numStudents;

cout << "Введіть кількість студентів: ";

cin >> numStudents;

Student\* students = new Student[numStudents];

int\* indexArray = new int[numStudents];

int menuItem;

do {

cout << "\nМеню:\n";

cout << "1 - Введення даних студентів\n";

cout << "2 - Вивід даних студентів\n";

cout << "3 - Фізичне сортування студентів\n";

cout << "4 - Індексне сортування студентів\n";

cout << "5 - Пошук студента\n";

cout << "0 - Вихід\n";

cout << "Виберіть дію: ";

cin >> menuItem;

switch (menuItem) {

case 1:

InputStudents(students, numStudents);

break;

case 2:

PrintStudents(students, numStudents);

break;

case 3:

PhysicalSort(students, numStudents);

cout << "Список студентів фізично відсортовано.\n";

break;

case 4:

IndexSort(students, numStudents, indexArray);

cout << "Список студентів індексно відсортовано.\n";

break;

case 5: {

string surname;

int course, profileGrade;

cout << "Введіть прізвище для пошуку: ";

cin >> surname;

cout << "Введіть курс для пошуку: ";

cin >> course;

cout << "Введіть оцінку з профільного предмету для пошуку: ";

cin >> profileGrade;

int result = BinarySearch(students, indexArray, numStudents, surname, course, profileGrade);

if (result != -1)

cout << "Студент знайдений за індексом: " << result << "\n";

else

cout << "Студент не знайдений.\n";

break;

}

case 0:

break;

default:

cout << "Неправильний вибір. Спробуйте ще раз.\n";

}

} while (menuItem != 0);

delete[] students;

delete[] indexArray;

return 0;

}

void InputStudents(Student\* students, int N) {

int spec;

for (int i = 0; i < N; i++) {

cout << "Студент № " << i + 1 << ":\n";

cin.ignore();

cout << " Прізвище: "; getline(cin, students[i].surname);

cout << " Курс: "; cin >> students[i].course;

cout << " Спеціальність (0 - Комп'ютерні науки, 1 - Інформатика, 2 - Математика та економіка, 3 - Фізика та інформатика, 4 - Трудове навчання): ";

cin >> spec;

students[i].specialization = static\_cast<Specialization>(spec);

cout << " Оцінка з фізики: "; cin >> students[i].physics;

cout << " Оцінка з математики: "; cin >> students[i].mathematics;

if (students[i].specialization == COMPUTER\_SCIENCE) {

cout << " Оцінка з програмування: ";

cin >> students[i].programming;

}

else if (students[i].specialization == INFORMATICS) {

cout << " Оцінка з чисельних методів: ";

cin >> students[i].numerical\_methods;

}

else {

cout << " Оцінка з педагогіки: ";

cin >> students[i].pedagogy;

}

}

}

void PrintStudents(const Student\* students, int N) {

cout << "=========================================================================================================\n";

cout << "| № | Прізвище | Курс | Спеціальність | Фізика | Математика | Профільний предмет |\n";

cout << "---------------------------------------------------------------------------------------------------------\n";

for (int i = 0; i < N; i++) {

int profileGrade = GetProfileGrade(students[i]);

cout << setw(2) << left << "| " << i + 1 << " | " << setw(18) << students[i].surname << " | " << setw(4) << students[i].course << " | ";

cout << setw(21) << left << specializationStr[students[i].specialization] << " | " << setw(6) << students[i].physics << " | ";

cout << setw(11) << students[i].mathematics << " | " << setw(20) << profileGrade << " |\n";

}

cout << "=========================================================================================================\n";

}

int GetProfileGrade(const Student& student) {

if (student.specialization == COMPUTER\_SCIENCE) return student.programming;

else if (student.specialization == INFORMATICS) return student.numerical\_methods;

else return student.pedagogy;

}

void PhysicalSort(Student\* students, int N) {

sort(students, students + N, [](const Student& a, const Student& b) {

int gradeA = GetProfileGrade(a);

int gradeB = GetProfileGrade(b);

if (gradeA != gradeB) return gradeA > gradeB;

if (a.course != b.course) return a.course < b.course;

return a.surname > b.surname;

});

}

void IndexSort(const Student\* students, int N, int\* indexArray) {

for (int i = 0; i < N; i++) {

indexArray[i] = i;

}

sort(indexArray, indexArray + N, [&](int a, int b) {

int gradeA = GetProfileGrade(students[a]);

int gradeB = GetProfileGrade(students[b]);

if (gradeA != gradeB) return gradeA > gradeB;

if (students[a].course != students[b].course) return students[a].course < students[b].course;

return students[a].surname < students[b].surname;

});

}

int GetProfileGrade(const Student\* students, const int\* indexArray, int N, const string& surname, int course, int profileGrade) {

int left = 0, right = N - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

int idx = indexArray[mid];

const Student& student = students[idx];

int studentGrade = GetProfileGrade(student);

if (student.surname == surname && student.course == course && studentGrade == profileGrade) {

return idx;

}

else if (studentGrade < profileGrade || (studentGrade == profileGrade && student.course > course) ||

(studentGrade == profileGrade && student.course == course && student.surname < surname)) {

right = mid - 1;

}

else {

left = mid + 1;

}

}

return -1;

}

Зображення, що містить знімок екрана, текст, Мультимедійне програмне забезпечення, програмне забезпечення

Автоматично згенерований опис

GitHub:

Висновки: На цій лабораторній я навчився впорядковувати масив структур з об’єднаннями. Навчитися здійснювати фізичне та індексне впорядкування. Навчився здійснювати бінарний пошук у фізично чи індексно впорядкованому масиві.