НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА

ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ



ЗВІТ

про виконання лабораторної роботи № 9,2

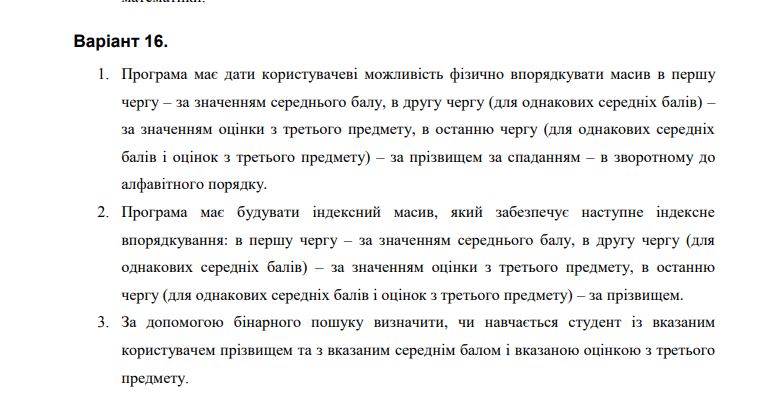
«Впорядкування та бінарний пошук в масиві структур»

з дисципліни «Алгоритмізація та програмування»

студентки групи ІТ-12

Прендкович Ірини Віталіївни

Умова завдання



Відповідь

#include <iostream>

#include <iomanip>

#include <string>

#define TESTING

using namespace std;

#pragma pack(push, 1)

enum Speciality { ITIS, RE, KN, KI, PZ };

string speciality\_str[] = { "Computer Science ",

"Radio Electronics", "Computer Science",

"Computer Engineering", "Software" };

struct Students

{

string surname;

int course;

Speciality speciality;

int grade\_physics;

int grade\_math;

int grade\_info;

int third\_grade;

};

#pragma pack(pop)

void Create(Students\* S, const int N);

double student\_avarage\_evaluation(Students\* S, const int i);

void Print(Students\* S, const int N);

double Percentage(Students\* S, const int N);

void p\_sort(Students\* S, const int N);

int\* i\_sort(Students\* S, const int N);

void i\_print(Students\* S, int\* I, const int N);

bool b\_search(Students\* S, const int N, const string surname, const double avarage\_evaluation, const int third\_grade);

#ifdef TESTING

int main()

{

int N;

cout << "Enter the number of students: "; cin >> N;

Students\* S = new Students[N];

string f\_surname;

double f\_avarage\_evaluation;

int f\_third\_grade;

int MenuItem;

while (true)

{

cout << endl << endl << endl;

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

cout << "1 - Enter information about students" << endl;

cout << "2 - Display information about students" << endl;

cout << "3 - Physically organize information" << endl;

cout << "4 - Index information" << endl;

cout << "5 - Student Search"; cout << endl;

cout << "0 - Shut down the program" << endl;

cout << "Enter:"; cin >> MenuItem; cout << endl;

switch (MenuItem)

{

case 1:

Create(S, N);

break;

case 2:

Print(S, N);

break;

case 3:

p\_sort(S, N);

Print(S, N);

break;

case 4:

i\_print(S, i\_sort(S, N), N);

break;

case 5:

cout << "Enter search criteria: " << endl;

cout << "surname" << endl;

cin.get();

cin.sync();

getline(cin, f\_surname); cout << endl;

do

{

cout << "GPA" << endl;

cin >> f\_avarage\_evaluation; cout << endl;

} while (f\_avarage\_evaluation < 1 || f\_avarage\_evaluation > 5);

do

{

cout << "Grade from the third subject:" << endl;

cin >> f\_third\_grade; cout << endl;

} while (f\_third\_grade < 1 || f\_third\_grade > 5);

if (b\_search(S, N, f\_surname, f\_avarage\_evaluation, f\_third\_grade))

{

cout << "Student is" << endl;

}

else {

cout << "not found" << endl;

}

}

if (MenuItem == 0) break;

}

return 0;

}

#endif

void Create(Students\* S, const int N)

{

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

{

int speciality;

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

cin.get();

cin.sync();

cout << "Surname: ";

getline(cin, S[i].surname); cout << endl;

do

{

cout << "Course: "; cin >> S[i].course; cout << endl;

} while (S[i].course < 1 || S[i].course > 6);

do

{

cout << "Specialty (0 - Informatics, 1 - Radio Electronics, 2 - Computer Science,";

cout << "\ n3 - Computer Engineering, 4 - Software) ";

cin >> speciality;

S[i].speciality = (Speciality)speciality;

} while (S[i].speciality < 0 || S[i].speciality > 4);

do

{

cout << "physics grade: "; cin >> S[i].grade\_physics; cout << endl;

} while (S[i].grade\_physics < 1 || S[i].grade\_physics > 5);

do

{

cout << "maths grade: "; cin >> S[i].grade\_math; cout << endl;

} while (S[i].grade\_math < 1 || S[i].grade\_math > 5);

do

{

cout << "programming grade: "; cin >> S[i].grade\_info; cout << endl;

} while (S[i].grade\_info < 1 || S[i].grade\_info > 5);

cout << endl;

}

}

void Print(Students\* S, const int N)

{

cout << "=========================================================================================" << endl;

cout << "| # | Last name | Course | Specialty | Physics |Mathematics | Programming |" << endl;

cout << "-----------------------------------------------------------------------------------------" << endl;

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

{

cout << "|" << setw(3) << i + 1 << setw(3);

cout << "|" << setw(4) << S[i].surname << setw(4);

cout << "|" << setw(6) << S[i].course << setw(6);

cout << "|" << setw(6) << speciality\_str[S[i].speciality] << setw(6);

cout << "|" << setw(6) << S[i].grade\_physics << setw(6);

cout << "|" << setw(6) << S[i].grade\_math << setw(6);

cout << "|" << setw(6) << S[i].grade\_info << setw(6) << "|" << endl;

}

cout << "=========================================================================================" << endl << endl;

}

double student\_avarage\_evaluation(Students\* S, int i)

{

return (S[i].grade\_math \* 1. + S[i].grade\_physics \* 1. + S[i].grade\_info \* 1.) / 3.;

}

void p\_sort(Students\* S, int N)

{

Students temp;

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

{

for (int j = 0; j < N - i - 1; j++)

{

if ((student\_avarage\_evaluation(S, j) > student\_avarage\_evaluation(S, j + 1))

|| (student\_avarage\_evaluation(S, j) == student\_avarage\_evaluation(S, j + 1))

&& (S[j].third\_grade > S[j + 1].third\_grade)

|| (student\_avarage\_evaluation(S, j) == student\_avarage\_evaluation(S, j + 1))

&& (S[j].third\_grade == S[j + 1].third\_grade)

&& (S[j].surname < S[j].surname))

{

temp = S[j];

S[j] = S[j + 1];

S[j + 1] = temp;

}

}

}

}

int\* i\_sort(Students\* S, const int N)

{

int\* I = new int[N];

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

I[i] = i;

int j, value;

for (int i = 1; i < N; i++)

{

value = I[i];

for (j = i - 1;

(student\_avarage\_evaluation(S, j) > student\_avarage\_evaluation(S, j + 1))

|| (student\_avarage\_evaluation(S, j) == student\_avarage\_evaluation(S, j + 1))

&& (S[j].third\_grade > S[j + 1].third\_grade)

|| (student\_avarage\_evaluation(S, j) == student\_avarage\_evaluation(S, j + 1))

&& (S[j].third\_grade == S[j + 1].third\_grade)

&& (S[j].surname < S[j].surname);

j--)

{

I[j + 1] = I[j];

}

I[j + 1] = value;

}

return I;

}

void i\_print(Students\* S, int\* I, const int N)

{

cout << "=========================================================================================" << endl;

cout << "| # | Last name | Course | Specialty | Physics |Mathematics | Programming |" << endl;

cout << "-----------------------------------------------------------------------------------------" << endl;

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

{

cout << "|" << setw(3) << i + 1 << setw(3);

cout << "|" << setw(4) << S[I[i]].surname << setw(4);

cout << "|" << setw(6) << S[I[i]].course << setw(6);

cout << "|" << setw(6) << speciality\_str[S[I[i]].speciality] << setw(6);

cout << "|" << setw(6) << S[I[i]].grade\_physics << setw(6);

cout << "|" << setw(6) << S[I[i]].grade\_math << setw(6);

cout << "|" << setw(6) << S[I[i]].grade\_info << setw(6) << "|" << endl;

}

cout << "=========================================================================================" << endl << endl;

}

bool b\_search(Students\* S, const int N, const string surname, const double avarage\_evaluation, const int third\_grade)

{

int L = 0, R = N - 1, m;

do

{

m = (L + R) / 2;

if ((S[m].surname == surname

&&

(student\_avarage\_evaluation(S, m) == avarage\_evaluation)

&&

(S[m].third\_grade == third\_grade)))

{

return 1;

}

if ((S[m].third\_grade < third\_grade)

||

(S[m].third\_grade == third\_grade

&&

student\_avarage\_evaluation(S, m) < avarage\_evaluation)

||

(S[m].third\_grade == third\_grade

&&

student\_avarage\_evaluation(S, m) == avarage\_evaluation

&&

S[m].surname < surname))

{

L = m + 1;

}

else

{

R = m - 1;

}

} while (L <= R);

return 0;

}

#include "pch.h"

#include "CppUnitTest.h"

#include "../lab 9.2 A/lab 9.2 A.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest92A

{

TEST\_CLASS(UnitTest92A)

{

public:

TEST\_METHOD(TestMethod1)

{

Students\* S = new Students[1];

S->surname = "Petr";

S->grade\_info = 5;

S->grade\_math = 4;

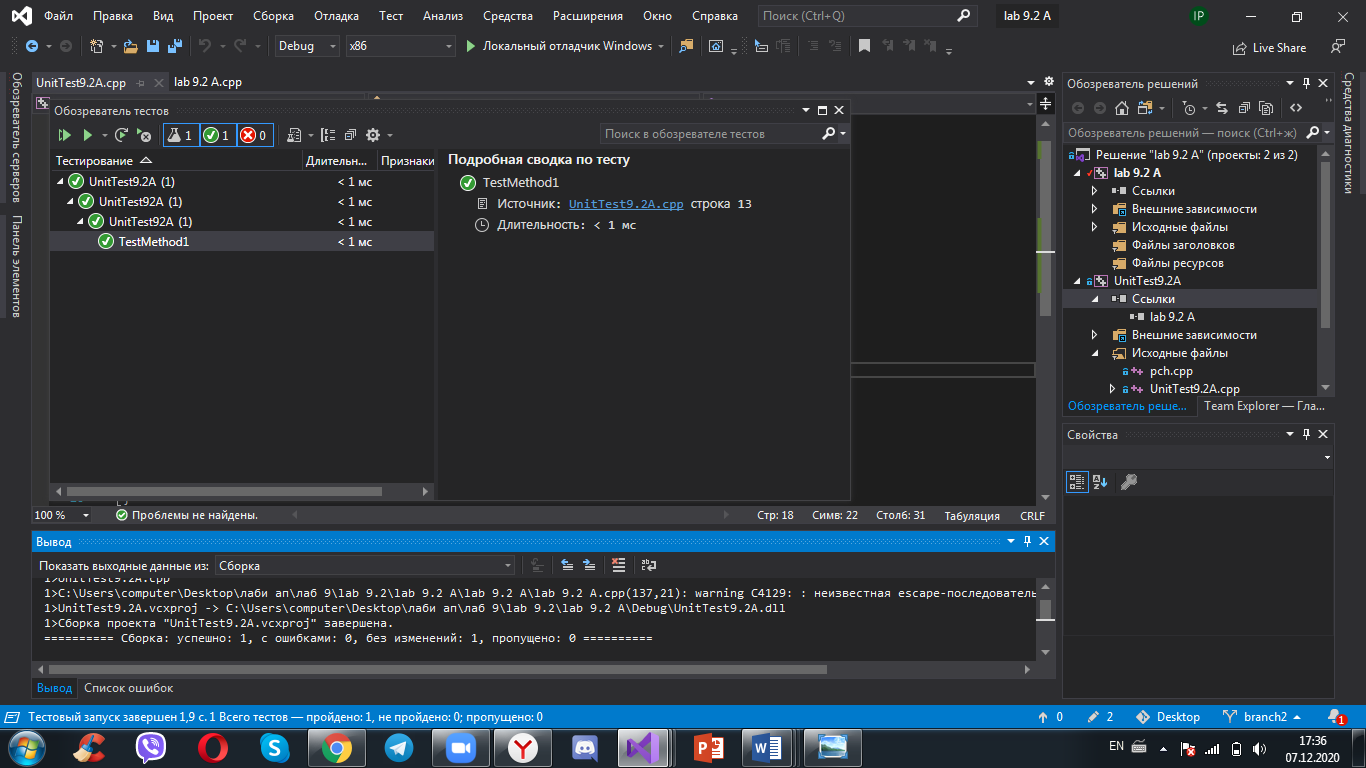
S->grade\_physics = 3;

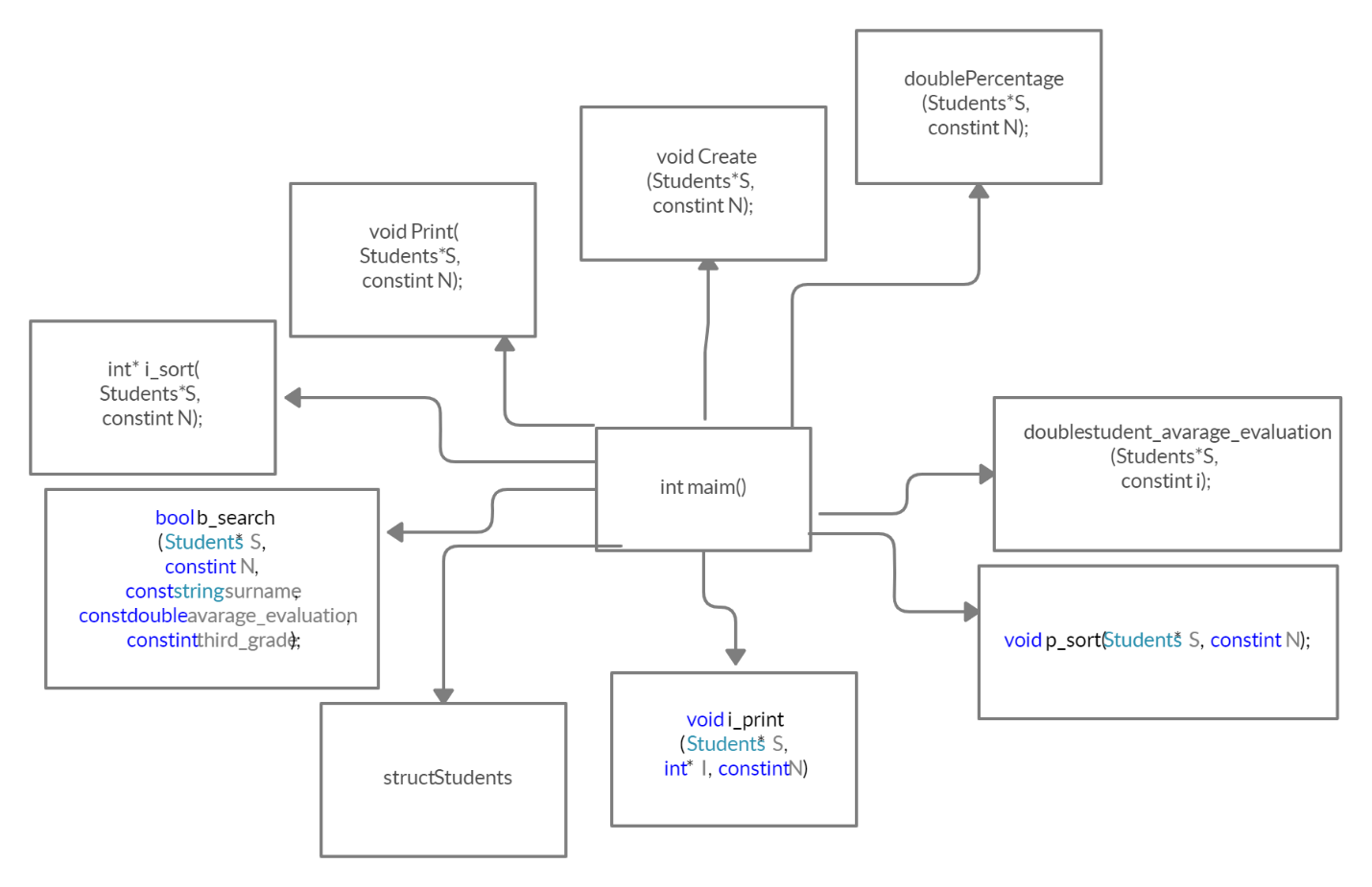
bool functionResult = b\_search(S, 1, "Petr", 4, 4);

Assert::AreEqual(false, functionResult);

}

};

}



<https://github.com/irynaprendkovychitis2020/lab-9.2>

Висновок

На цій лабораторній я навчилася впорядковувати масив структур з об’єднаннями, здійснювати фізичне та індексне впорядкування, здійснювати бінарний пошук у фізично чи індексно впорядкованому масиві.