**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 8

**На тему:** *“Структури та об’єднання”*

**З дисципліни:** *“Основи програмування”*

**Лектор:**

ст.викл. каф. ПЗ

Муха Т.О.

**Виконав:**

ст. гр. ПЗ-14

Губик А. С.

**Прийняв:**

ст.викл. каф ПЗ

Кутельмах Р. К.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_ .

Львів – 2022

**Тема роботи:** структури та об’єднання.

**Мета роботи:** навчитися створювати нові типи даних у вигляді структур та об’єднань, а

також розробляти алгоритми їх обробки засобами мови С.

**Індивідуальне завдання**

З текстового файлу зчитати послідовність записів, які містять дані про результати сесії

студентів групи у такому форматі: <Прізвище>, <Ім’я>, <Дата народження>, <Список

екзаменаційних оцінок>. Роздрукувати введені дані у вигляді таблиці, а також подати

інформацію згідно варіанту.

Відсортувавати дані за віком студентів у зростаючому порядку. Роздрукувати список

студентів з рейтинговим балом нижчим від середнього балу в групі.

**Теоретичні відомості**

2.1. Загальне поняття про структури

Структури – це особливий комбінований тип даних, який об’єднує у спільне ціле набір логічно пов’язаних між собою різнотипних компонентів. Складові частини структури називають полями або елементами. Кожне поле структури має свій тип і своє ім’я:

struct ім'я\_структури

{

тип1 ім’я поля 1;

тип2 ім’я поля 2;

.

.

.

типN ім’я поля N;

};

Наприклад:

struct student

{

char surname[50]

char name[50];

int year;

};

Якщо в програмі використовується тільки одна структура , то її шаблон можна оголосити без імені структури (тега):

struct {

список полів структури;

};

Такий шаблон називають безіменним. Можна залишити без іменним один з усіх оголошених в програмі шаблонів структур. Кількість і склад полів структури визначається самим програмістом. Як правило, структурами представляються взаємозв’язані дані.

Наприклад, структури широко застосовуються для представлення часу та дати (день, місяць, рік, година, хвилина, секунда).

Поля структури описуються послідовно. В оперативній пам’яті вони будуть записані саме в тому порядку, в якому були задані в шаблоні даної структури. Тип поля може бути довільним простим або складеним типом. Ім’я поля є ідентифікатором змінної відповідного типу. В межах одного шаблону імена всіх полів повинні бути різними, але вони можуть збігатись з іменами полів інших структур чи з іменами змінних програми.

Слід пам’ятати, що описана вище конструкція не створює нової змінної і не виділяє пам'ять. Це просто логічна схема! Застосовують дві форми оголошення структурних змінних:

 Через посилання на попередньо оголошений шаблон;

 Одночасно з оголошенням шаблону структури.

Тобто щоб створити один чи більше екземплярів структури , слід після закриваючої фігурної дужки в її оголошенні перелічити імена екземплярів, розділені комами. Інший варіант – слід після імені структури вказати ім’я її екземпляра (так само, як створюються змінні, скажімо, типу int).

2.2. Вказівники на структури

Оскільки структурні змінні не є вказівниками, то до них можна застосовувати операцію визначення адреси &. Результатом операції буде адреса першого байта ділянки, яку займає в оперативній пам’яті структура-операнд. Значення адреси структури можна присвоїти

вказівнику, базовий тип якої збігається з типом структурної змінної. Якщо оголошено структуру

struct point

{

int x;

int y;

int z;

} point1 = {-1,16,5};

то вказівник на неї виглядатиме так:

struct point \*pnt;

Щоб присвоїти вказівнику pnt адресу екземпляра point1 структури point, слід записати:

pnt = &point1;

Доступ до полів структури через вказівник здійснюється за допомогою оператора ->.

Вказівник на структуру->Ім’я поля.

printf("%d", pnt->x);

printf("%d", pnt->y);

printf("%d", pnt->z);

2.3. Вкладені структури

Структура може бути елементом структури іншого типу. Для вкладених структур діють ті ж правила оголошення, як і для зовнішніх: можна окремо оголошувати шаблон структури або задавати його всередині зовнішньої структури разом з оголошенням відповідних змінних.

Приклад вкладення структур:

struct student{

char surname[50];

char name[50];

int year;

};

struct marks

{

int mark[10];

};

struct ComplexStructure

{

struct student personal;

struct marks allmarks;

};

struct ComplexStructure item = {{"Ivanenko","Ivan",1},

{71,88,92,76,80,82,74,79,93,90}};

printf("%s", item.personal.surname);

printf("%d", item.allmarks.mark[4]);char surname[50];

Екземпляри структур student та marks є полями структури ComplexStructure. Слід зауважити, що екземпляр item структури ComplexStructure не можна було б ініціалізувати екземплярами структур student та marks – це б призвело до помилки.

Доступ до кожної з вкладених структур здійснюється через оператор “.”, як і до будь- якого поля структури, і доступ до полів вкладених структур виконується, у свою чергу, за допомогою оператора “.”.

Можна організувати змішаний доступ до полів структури – до самих вкладених структур через вказівники, а до їхніх полів – за допомогою оператора “.”.

**Код програми**

Назва файлу: **sort.h**

void SortArray(struct student arr[], int size,

int (\*condition)(struct student\*, struct student\*));

int AgeUp(struct student \*a, struct student \*b);

Назва файлу: **table.h**

void ReadList(FILE \*f, struct student arr[], int size);

void PrintTable(FILE \*f, const struct student arr[], int size,

int (\*condition)(const struct student\*, float), float n);

float AverageRating(const struct student arr[], int size);

int isGreaterThanAverageRating(const struct student \*s,

float avr);

int True(const struct student \*s, float avr);

Назва файлу: **students.h**

struct ExamGrade{

int OP;

int CDM;

int Calculus;

int LAAG;

int History;

};

struct student{

char surname[100];

char name[100];

char birthdate[100];

struct ExamGrade ExamGrade;

};

struct item{

struct student student;

struct item \*next;

};

Назва файлу: **table.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "student.h"

#include "table.h"

#define SURNAME\_SIZE 14

#define NAME\_SIZE 10

void PrintSpaces(FILE \*f, int n)

{

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

fprintf(f, " ");

}

void PrintRow(FILE \*f, const struct student \*s, int n)

{

fprintf(f, "| %02d ", n);

fprintf(f, "| %s", s->surname);

PrintSpaces(f, SURNAME\_SIZE - strlen(s->surname) / 2 - 1);

fprintf(f, "| %s", s->name);

PrintSpaces(f, NAME\_SIZE - strlen(s->name) / 2 - 1);

fprintf(f, "| %s ", s->birthdate);

fprintf(f, "| %02d ", s->ExamGrade.OP);

fprintf(f, "| %02d ", s->ExamGrade.CDM);

fprintf(f, "| %02d ", s->ExamGrade.Calculus);

fprintf(f, "| %02d ", s->ExamGrade.LAAG);

fprintf(f, "| %02d |", s->ExamGrade.History);

fprintf(f, "\n");

}

void PrintTable(FILE \*f, const struct student arr[], int size,

int (\*condition)(const struct student\*, float), float n)

{

fprintf(f, "---------------------------------------------"\

"-------------------------------\n");

fprintf(f, "| | | | дата |"\

" Список екзаменаційних оцінок |\n");

fprintf(f, "| № | Прізвище | Ім'я | |"\

"------------------------------|\n");

fprintf(f, "| | | | народження |"\

" ОП | КДМ | MA | ЛААГ | Іст. |\n");

fprintf(f, "---------------------------------------------"\

"-------------------------------\n");

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

if (condition(&arr[i], n))

PrintRow(f, &arr[i], i + 1);

fprintf(f, "---------------------------------------------"\

"-------------------------------\n");

}

float AverageRating(const struct student arr[], int size)

{

float avr = 0;

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

float sum = 0;

sum += arr[i].ExamGrade.OP;

sum += arr[i].ExamGrade.CDM;

sum += arr[i].ExamGrade.Calculus;

sum += arr[i].ExamGrade.LAAG;

sum += arr[i].ExamGrade.History;

avr += sum / 5;

}

return avr / 27;

}

int isGreaterThanAverageRating(const struct student \*s, float avr)

{

float n = 0;

n += s->ExamGrade.OP;

n += s->ExamGrade.CDM;

n += s->ExamGrade.Calculus;

n += s->ExamGrade.LAAG;

n += s->ExamGrade.History;

n /= 5;

//printf("%lf %lf\n", avr, n);

if (n < avr)

return 1;

return 0;

}

int True(const struct student \*s, float avr)

{

return 1;

}

void ReadList(FILE \*f, struct student arr[], int size)

{

char str[200] = "";

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

fgets(str, 200, f);

char \*token = strtok(str, " \n");

strcpy(arr[i].surname, token);

token = strtok(NULL, " \n");

strcpy(arr[i].name, token);

token = strtok(NULL, " \n");

strcpy(arr[i].birthdate, token);

token = strtok(NULL, " \n");

arr[i].ExamGrade.OP = atoi(token);

token = strtok(NULL, " \n");

arr[i].ExamGrade.CDM = atoi(token);

token = strtok(NULL, " \n");

arr[i].ExamGrade.Calculus = atoi(token);

token = strtok(NULL, " \n");

arr[i].ExamGrade.LAAG = atoi(token);

token = strtok(NULL, " \n");

arr[i].ExamGrade.History = atoi(token);

token = strtok(NULL, " \n");

}

}

Назва файлу: **sort.c**

#include <string.h>

#include <stdlib.h>

#include "student.h"

#include "sort.h"

void swap(struct student \*a, struct student \*b)

{

struct student tmp = \*a;

\*a = \*b;

\*b = tmp;

}

void SortArray(struct student arr[], int size,

int (\*condition)(struct student\*, struct student\*))

{

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

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

if(condition(&arr[j], &arr[j + 1]))

swap(&arr[j], &arr[j + 1]);

}

int AgeUp(struct student \*a, struct student \*b)

{

char strA[100];

char strB[100];

strcpy(strA, a->birthdate);

strcpy(strB, b->birthdate);

char \*token = strtok(strA, ".");

int age1 = 10 \* atoi(token);

token = strtok(NULL, ".");

age1 += 100 \* atoi(token);

token = strtok(NULL, ".");

age1 += 10000 \* atoi(token);

token = strtok(strB, ".");

int age2 = 10 \* atoi(token);

token = strtok(NULL, ".");

age2 += 100 \* atoi(token);

token = strtok(NULL, ".");

age2 += 10000 \* atoi(token);

if(age1 < age2)

return 1;

return 0;

}

Назва файлу: **main.c**

#include <stdio.h>

#include "student.h"

#include "table.h"

#include "sort.h"

#define LIST\_SIZE 27

int main()

{

FILE \*f;

struct student arr[LIST\_SIZE];

f = fopen("list.txt", "r");

ReadList(f, arr, LIST\_SIZE);

fclose(f);

//PrintTable(stdout, arr, LIST\_SIZE, True, 0);

f = fopen("unsorted.txt", "w");

PrintTable(f, arr, LIST\_SIZE, True, 0);

fclose(f);

SortArray(arr, LIST\_SIZE, AgeUp);

f = fopen("sorted.txt", "w");

PrintTable(f, arr, LIST\_SIZE, True, 0);

fclose(f);

float avr = AverageRating(arr, LIST\_SIZE);

f = fopen("SortedWithRating.txt", "w");

PrintTable(f, arr, LIST\_SIZE, isGreaterThanAverageRating, avr);

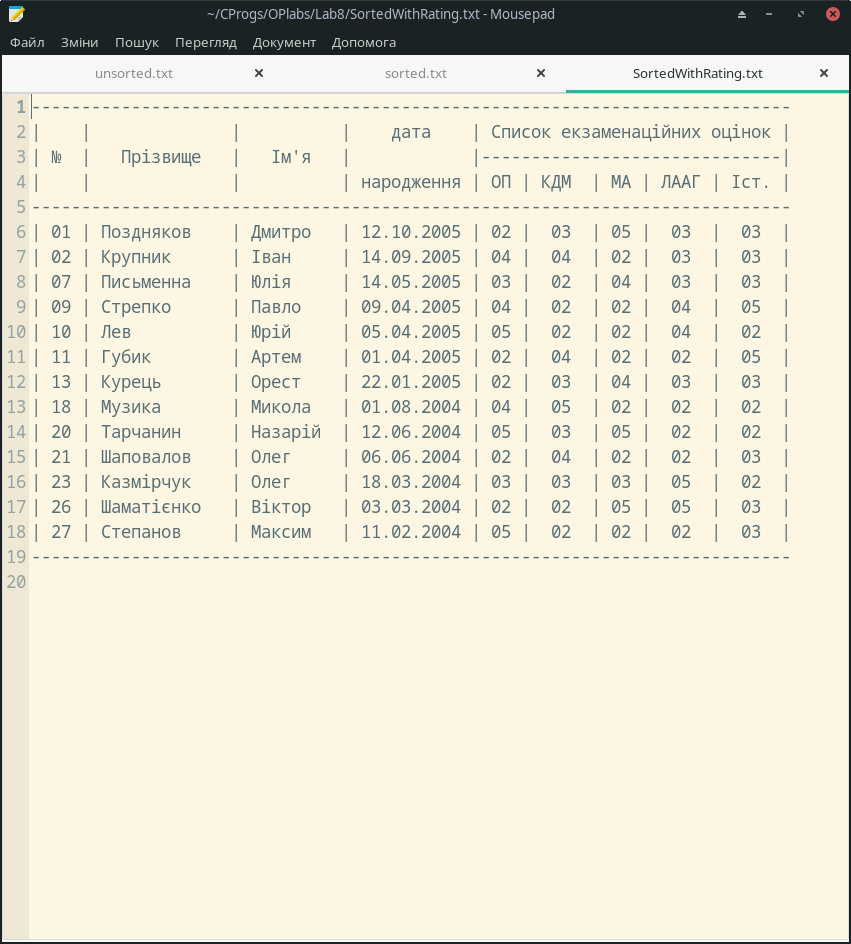
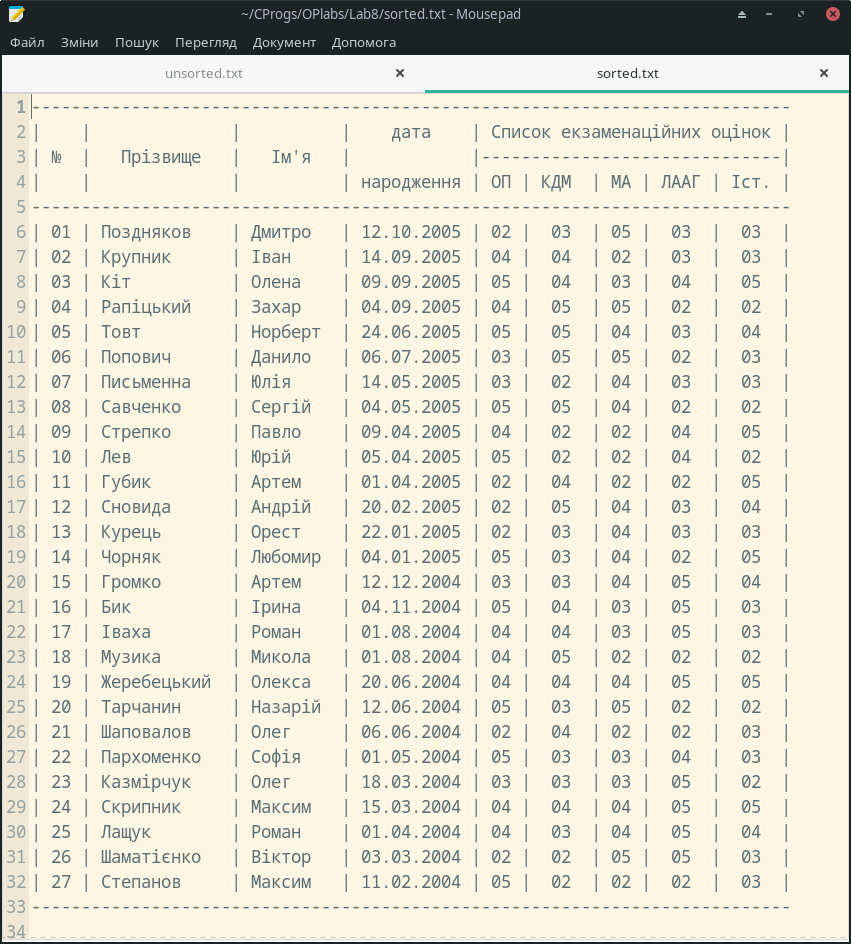
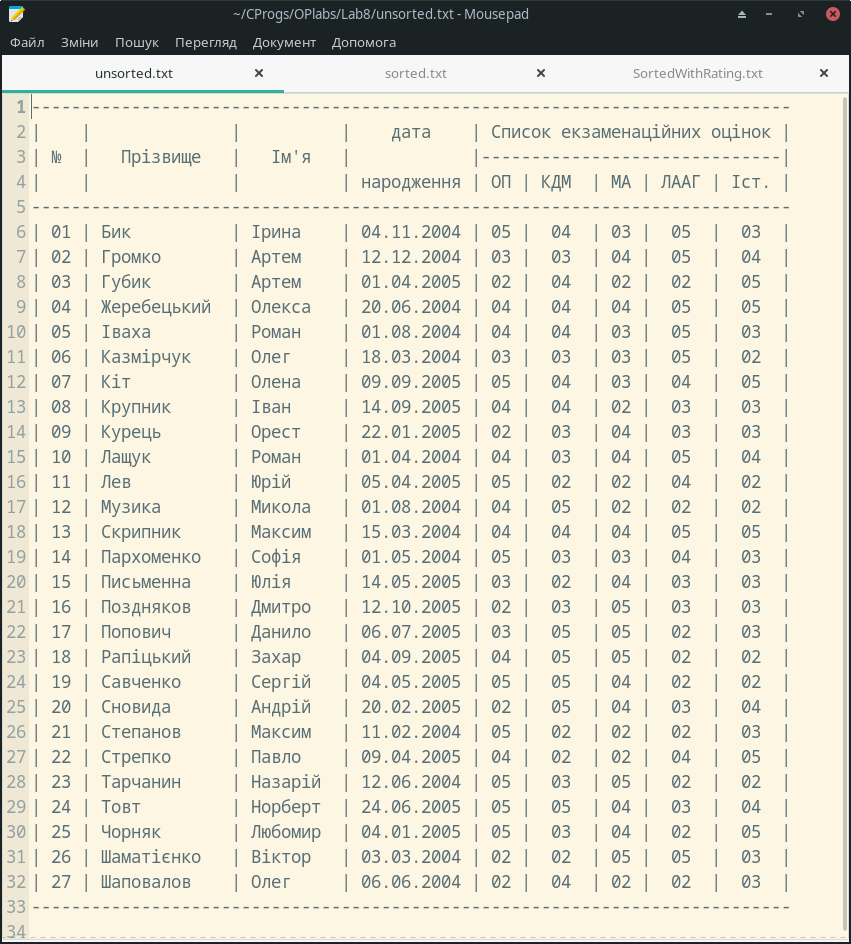
fclose(f);

return 0;

}

**Протокол роботи**

**main.c**



**Висновки**

Структури дають змогу впорядкувати дані, щоб організувати доступ до них і полегшити їх використання.