# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**Кафедра програмного забезпечення систем**

**ЗВІТ**

**Про виконання лабораторної роботи №7**

**«Структури та об’єднання»**

**з дисципліни «Основи програмування»**

Виконав:

слухач групи ПЗС-11

Гринчук Тарас

Прийняв:

доц. Макар В.М.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2013 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2013

**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++.

## 1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1. Структура являє собою сукупність різнотипних змінних, об’єднаних одним іменем. Змінні, що входять до складу структури прийнято називати полями структури. Оголошення структури здійснюється наступним чином:

*struct ім'я\_структури*

*тип1 поле, …, поле;*

*тип2 поле, …, поле;*

*………………………..*

*типN поле, …, поле;*

*};*

1. 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. Доступ до полів структури можна здійснити за допомогою оператора “.”. Наприклад:

*strcpy(student3.name,"Name");*

*strcpy(student3.surname,"LastName");*

*student3.year = 1;*

**Приклад програми (результат на рис. 1.1):**

#include <stdio.h>

#include <iostream>

#include <conio.h>

#include <string.h>

using namespace std;

int main ()

{

const int N = 5;

struct student{

char surname[50];

char name[50];

int mark;

};

student PI1[N]; /\* PI1 є масивом з 5 елементів типу student \*/

/\* В циклі вводимо всі елементи масиву з клавіатури \*/

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

cout<<"Enter the surname of "<<(i+1)<<"-th student"<<endl;

cin>>PI1[i].surname;

cout<<"Enter the name of "<<(i+1)<<"-th student"<<endl;

cin>>PI1[i].name;

cout<<"Enter the mark of "<<(i+1)<<"-th student"<<endl;

cin>>PI1[i].mark;

}

int max = 0, indexOfMax = -1;

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

if(PI1[i].mark > max) { max = PI1[i].mark; indexOfMax = i;}

}

cout<<"A sudent with the highest mark "<<PI1[indexOfMax].mark<<" is"<<PI1[indexOfMax].name<<" "<<PI1[indexOfMax].surname;

\_getch();

return 0;

}

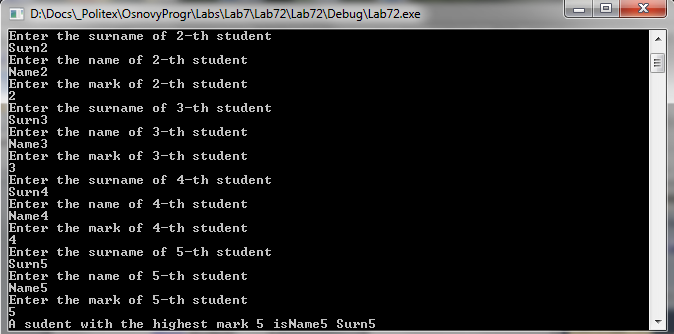


Рис. 1.1

1. Структура може бути елементом структури іншого типу. При цьому вкладення структур однакового типу не допускається. Приклад вкладення структур:

*struct student{*

*char surname[50], name[50];*

*int year;*

*}*

*struct marks*

*{*

*int mark[10];*

*};*

*struct ComplexStructure*

*struct student personal;*

*struct marks allmarks;*

*};*

*ComplexStructure item =*

*{{"Ivanenko","Ivan",1},{71,88,92,76,80,82,74,79,93,90}};*

*cout<<item.personal.surname<<endl;*

*cout<<item.allmarks.mark[4]<<endl;*

Екземпляри структур student та marks є полями структури ComplexStructure. 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 “.”. Наприклад:

*ComplexStructure \*p;*

*p = &item;*

*cout<<p->personal.name<<endl;*

*cout<<p->allmarks.mark[5]<<endl;*

1. Об’єднання, як і структура, дозволяє зберігати пов’язані дані всередині однієї змінної. Однак, розмір екземпляра структури рівний сумі розмірів усіх її полів, а розмір об’єднання рівний розміру найбільшого елемента, що входить у це об’єднання. Об’єднання застосовуються для більш економного використання пам’яті. Об’єднання створюється наступним чином:

*union ім’я\_об’єднання*

*{*

*тип ім’я\_елемента1;*

*тип ім’я\_елемента2;*

*тип ім’я\_елементаN*

*} імена\_екземплярів\_об’єднання;*

1. Якщо в екземплярі структури зберігаються усі її поля, то об’єднання зберігає лише один з елементів у кожен момент часу. Присвоєння значення іншому елементу об’єднання означає, що попереднє значення буде втрачене.

**Приклад (результат виконання на рис. 1.2):**

#include <iostream>

#include <conio.h>

using namespace std;

int main()

{

union geomObject

{

int oznaka:1;

struct twoDimensional

{

double x;

double y;

}object2D;

struct threeDimensional

{

double x;

double y;

double z;

}object3D;

} geom1;

geom1.oznaka = 0;

if(geom1.oznaka)

{

cout<<"Enter x"<<endl;

cin>>geom1.object3D.x;

cout<<"Enter y"<<endl;

cin>>geom1.object3D.y;

cout<<"Enter z"<<endl;

cin>>geom1.object3D.z;

}

else

{

cout<<"Enter x"<<endl;

cin>>geom1.object2D.x;

cout<<"Enter y"<<endl;

cin>>geom1.object2D.y;

}

cout<<geom1.object2D.x<<endl;

cout<<geom1.object2D.y<<endl;

geom1.oznaka = 1;

if(geom1.oznaka)

{

cout<<"Enter x"<<endl;

cin>>geom1.object3D.x;

cout<<"Enter y"<<endl;

cin>>geom1.object3D.y;

cout<<"Enter z"<<endl;

cin>>geom1.object3D.z;

}

else

{

cout<<"Enter x"<<endl;

cin>>geom1.object2D.x;

cout<<"Enter y"<<endl;

cin>>geom1.object2D.y;

}

cout<<geom1.object2D.x<<endl;

cout<<geom1.object2D.y<<endl;

\_getch();

return 0;

}

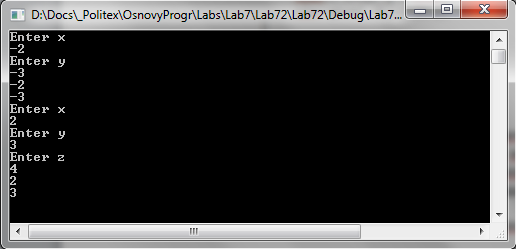


Рис. 1.2

1. Заголовочний файл fstream.h визначає клас вихідного файлового потоку з іменем ofstream. Використовуючи об’єкти класу ofstream, можна виводити вивід у файл. Для цього слід оголосити об’єкт типу ofstream, вказавши ім’я потрібного файлу як символьний рядок.

*ofstream file\_object("FILENAME.EXT");*

Для читання зі вхідного файлового потоку використовують об’єкти типу ifstream. Знову ж, потрібно створити об’єкт цього типу, передавши йому ім’я потрібного файлу:

*ifstream input\_file("filename.EXT");*

Для зчитування даних з текстового файлу може бути використана функція getline:

*char str1[64], str2[64], str3[64] ;*

*input\_file.getline(str1, sizeof(str1)) ; /\* тут послідовно зчитуються*

*три рядки з файлу у змінні str1, str2, str3\*/*

*input\_file.getline(str2, sizeof(str2));*

*input\_file.getline(str3, sizeof(str3)) ;*

Часто буває потрібно прочитати весь файл. Для цього організовується цикл while, умовою продовження якого є те, що ще не досягнуто кінець файлу. Зазначена умова перевіряється функцією eof(), яка повертає значення 0, якщо кінець файлу ще не зустрівся, і 1 – у протилежному випадку:

*while (!input\_file.eof())*

*{*

*// Оператори*

*}*

Після завершення роботи з файлом його потрібно закрити. Операційна система закриває файли після завершення роботи програми, однак хорошим тоном є виклик функції close у програмі:

input\_file.close ();

**Приклад (Результат на рис. 1.3):**

#include <iostream>

#include <fstream>

#include <conio.h>

#include <string.h>

void main(void)

{

using namespace std;

ifstream file1("student.txt"); /\* файл student.txt повинен

існувати, причому припускається, що кожен рядок у цьому файлі

містить інформацію про одного студента і окремі дані розділені

символами табуляції \*/

char str[200]; /\* у цю змінну будемо зчитувати текстовий файл по

рядках. Вважаємо, що довжина одного рядка файлу не перевищує 200

символів \*/

while (!file1.eof())/\* поки не досягнуто кінець файлу

student.txt\*/

{

file1.getline(str, sizeof(str)); /\* зчитуємо рядок файлу у

змінну str; курсор пересувається на наступний рядок\*/

cout<<str<<endl;

char \*pw; /\* розділяємо зчитаний рядок на окремі дані \*/

pw = strtok (str, "\t");

while(pw !=NULL)

{

cout << pw << endl;

pw = strtok (NULL, "\t");

}

}

\_getch();

}

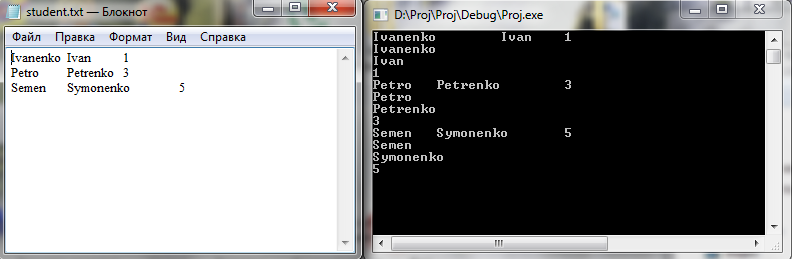


Рис. 1.3

## ХІД РОБОТИ

**Індивідуальне завдання (варіант 5).**

Програмна реалізація даного алгоритму має вигляд:

#include <iostream>

#include <fstream>

#include <sstream>

#include <conio.h>

#include <string.h>

#include <string>

using namespace std;

const int M=3;

const int N=5;

int strsort(char[20], char[20]);

struct student

{

char surname[20], name[20];

int date;

int marks[M];

};

void main(void) {

FILE\* file1 = fopen("stud.txt", "r");

if (!file1) {

cerr << "Error reading file !" << endl;

\_getch();

exit(1);

}

int i = 0,k;

student stud[30],temp;

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

fscanf (file1,"%s %s %d %d %d %d\n", &stud[i].surname, &stud[i].name, &stud[i].date, &stud[i].marks[0], &stud[i].marks[1], &stud[i].marks[2]);

cout<<stud[i].surname<<"\t"<<stud[i].name<<"\t"<<stud[i].date<<"\t"<<stud[i].marks[0]<<"\t"<<stud[i].marks[1]<<"\t"<<stud[i].marks[2]<<"\n";

}

cout<<endl;

//Сортування "методом бульбашки"

for(k=N-2; k>=0; k--)

for(i=0; i<=k; i++)

/\*Якщо i-те прізвище по алфавіту знаходиться пізніше за наступне – переставляєм\*/

if(strsort(stud[i].surname,stud[i+1].surname)){

temp=stud[i];

stud[i]=stud[i+1];

stud[i+1]=temp;

}

cout<<"Students with 2 grades\n";

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

int notfound=1;

for(int j=0; j<M; j++)

if(stud[i].marks[j]==2) {

notfound=0; break; //Знайшли першу двійку - перериваємо пошук

}

if(notfound) continue; /\*В цього студента нема двійок - перевіряємо наступного\*/

cout<<stud[i].surname<<"\t"<<stud[i].name<<"\t"<<stud[i].date<<"\t"<<stud[i].marks[0]<<"\t"<<stud[i].marks[1]<<"\t"<<stud[i].marks[2]<<"\n";

}

cout<<endl;

\_getch();

}

//Якщо перша строка по алфавіту знаходиться пізшіше - то 1, інакше - 0

int strsort(char s1[20], char s2[20]){

for(int i=0; i<20; i++) {

//букви одинакові - беремо наступну

if(s1[i]==s2[i]) continue;

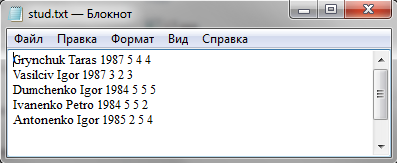
return (s1[i]>s2[i]);

}

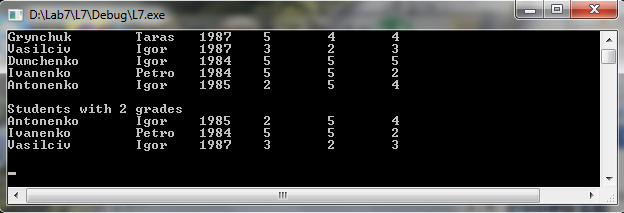
return 0;

}

Вхідний файл зображено на рис. 2.1, результати виконання програми на рис. 2.2.



(Рис. 2.1)



(Рис. 2.2)

## ВИСНОВКИ

На даній лабораторній роботі я навчився створювати нові типи даних у вигляді структур та об’єднань, а також розробляти алгоритми їх обробки засобами мови С++. Отримав знання по роботі з текстовими файлами у С++.