МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет “Львівська політехніка”

СТРУКТУРИ ТА ОБ’ЄДНАННЯ

ІНСТРУКЦІЯ
до лабораторної роботи № 8 з курсу
“Основи програмування”
для базового напрямку “Програмна інженерія”

Затверджено
На засіданні кафедри
програмного забезпечення
Протокол № від

ЛЬВІВ – 2018

1. МЕТА РОБОТИ

Мета роботи – навчитися створювати нові типи даних у вигляді структур та об’єднань, а також розробляти алгоритми їх обробки засобами мови С.

2. 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). Наприклад:

```
struct student
{
     char surname[50]
     char name[50];
     int year;
} student1, student2;

struct student student3;
```

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

Ім’я структурної змінної. Ім’я поля.

Наприклад:

```
strcpy(student3.name,"Name");
strcpy(student3.surname,"LastName");
student3.year = 1;
```

Відповідним чином виконується і виведення полів структури на екран:

```
printf("%s",student3.name);
printf("%s",student3.surname);
```

В оголошеннях структурні змінні можна ініціалізувати, тобто присвоювати їх елементам початкові значення. Список значень, якими ініціалізується структура, вказується у фігурних дужках. Приклад ініціалізації структурної змінної:

```
struct student
{
     char surname[50]
     char name[50];
     int year;
} student1, student2 = {"Ivan", "Ivanenko",1};

struct student student1 = {"Petro", "Petrenko",42}
```

При цьому значення, якими ініціалізується структура записують в тій самій послідовності, в якій вказані поля у шаблоні структури та типи повинні бути сумісними з типами відповідних полів. Також можна виконувати часткову ініціалізацію:

```
     student student3;
     printf("Enter all what you know about student3");
     printf("Name?");
     scanf("%s",&student3.name);
     printf("Surname?");
     scanf("%s",&student3.surname);
     printf("Year of study?");
     scanf("%d",&student3.year);

     strcpy(student1.name,"Petro");
     strcpy(student1.surname,"Petrenko");
     student1.year = 1;

     printf("Student1 %s %s of %d
year"student1.name,student1.surname,student1.year);
     printf("Student2 %s %s of %d year"
student2.name,student2.surname,student2.year);
```

```
printf("Student3 %s %s of %d
year"student3.name,student3.surname,student3.year);
```

Екземпляри структур можна присвоювати один одному, але лише у тому випадку, коли вони мають однаковий шаблон, тобто займають однакові за обсягом ділянки в оперативній пам'яті. Наприклад:

```
student1 = student3;
```

При цьому всі поля екземпляра структури student1 стануть ідентичними полям екземпляра структури student3. Однак, для порівняння двох структур слід порівнювати окремо їхні відповідні поля. Наприклад:

```
if(!strcmp(student1.name,student2.name) && !
strcmp(student1.surname,student2.surname)&& student1.year==student2.year)
    printf("student1 and student2 are equal");
else printf("student1 and student2 are not equal");

if(!strcmp(student1.name,student3.name) && !
strcmp(student1.surname,student3.surname) &&
student1.year==student3.year)
    printf("student1 and student3 are equal");
else printf("student1 and student3 are not equal");
```

Полем структури може бути масив. Наприклад,

```
    struct studentMarks
    {
        char surname[50];
        char name[50];
        int year;
        int marks[10];
    } stud1 = {"Future","Programmer",2,{71,88,92,76,80,82,74,79,93,90}},
stud2;
    printf("%d",stud1.marks[3]);
    printf("%d",stud1.marks[8]);
```

Серед полів структури studentMarks є ім'я та прізвище студента, курс та масив оцінок marks. Екземпляр структури stud1 ініціалізується одразу при створенні. До цього масиву доступаємося так само, як і до всіх інших полів структури – через оператор ".". Доступ до елементів масиву здійснюється у звичний спосіб – за допомогою оператора []. При виконанні даного фрагмента коду на екран виводяться числа 76 та 93, оскільки саме вони є елементами масиву marks з індексами 3 та 8 (індексація масивів починається з 0).

Можна оголошувати масив структур – так само, як і масиви будь-якого вбудованого типу.

Оголошення структури повинно передувати оголошенню масиву структур. Для доступу до структури, що є елементом массиву, використовують її індекс. Наприклад:

```
struct student
{
    char surname[50];
    char name[50];
    int mark;
};
struct student PI1[N]; /* масив N елементів типу student */

scanf("%s", &PI1[0].name); /* запис рядка символів з клавіатури у поле
name структури, що є першим елементом (з індексом 0) масиву PI1 */
```

Приклад.

В наступній програмі вводиться масив з 5 елементів типу `student`. Цей тип є структурою, поля якої задають ім'я та прізвище студента та його оцінку. Вибирається студент з найвищим балом та його ім'я виводиться на екран.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>


int main ()
{
    const int N = 5;
    int i=0;
    struct student{
        char surname[50];
        char name[50];
        int mark;
    };
 struct student PI1[5];

    for(i=0;i<N;i++) {
        printf("Enter the surname of %d -th student",(i+1));
        scanf("%s",&PI1[i].surname);
        printf("Enter the name of %d -th student",(i+1));
        scanf("%s",&PI1[i].name);
        printf("Enter the mark of %d -th student",(i+1));
        scanf("%d",&PI1[i].mark);
    }

    int max = 0, indexOfMax = -1;
    for(i = 0; i< N; i++){
        if(PI1[i].mark > max) {
            max = PI1[i].mark;
            indexOfMax = i;
        }
    }
    printf("A sudent with the highest mark %d is %s
%s",PI1[indexOfMax].mark, PI1[indexOfMax].name, PI1[indexOfMax].surname);
    _getch();
    return 0;
}
```

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 ".". Наприклад:

```
struct ComplexStructure *p;
p = &item;
printf("%s",p->personal.name);
printf("%d",p->allmarks.mark[5]);
```

2.4. Об'єднання

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.

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.

Об'єднання створюється наступним чином:

```
union ім'я_об'єднання
{
     Тип1 ім'я_елемента1;
     Тип2 ім'я_елемента2;
     .
     типN ім'я_елементаN
} імена_екземплярів_об'єднання;
```

Різниця між структурою та об'єднанням

`struct angle1`
`{`
`  int grad;`
`  double rad;`
`} ang1;`
	Займає обсяг пам'яті
	`sizeof(int)+sizeof(double)`
	та в кожен момент часу зберігає значення обох полів – grad та rad

`union angle2`
`{`
`  int grad;`
`  double rad;`
`} ang2;`
	Займає обсяг пам'яті
	`sizeof(double)`
	та в кожен момент часу зберігає лише один елемент - або grad, або rad, залежно від того, який з них записувався останнім

Приклад. Створюється об'єднання `geomObject`, елементами якого є "ознака" типу int та дві структури. Перша структура (`object2D`) описує точку на площині, а друга (`object3D`) – у просторі. Створюється екземпляр `geom1` об'єднання `geomObject`. Спочатку елементу `oznaka` присвоюється значення 0. Тоді це значення перевіряється і в залежності від того, є воно нульовим чи ні, заповнюється одна зі структур, що є елементами об'єднання. Для нульового значення "ознаки" заповнюється структура `object2D`. Після цього на екран виводиться вміст структури `object2D`. Далі значенню `oznaka` присвоюється значення 1, після чого виконується вже інша гілка коду – заповнення структури `object3D`. І знову на екран виводиться вміст структури `object2D`. При виконанні даного коду видно, що екземпляр об'єднання geom1 перезаписується.

```
#include <stdio.h>
#include <conio.h>

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)
    {
        printf("Enter x");
        scanf("%lf",&geom1.object3D.x);
        printf("Enter y");
        scanf("%lf",&geom1.object3D.y);
        printf("Enter z");
        scanf("%lf",&geom1.object3D.z);
    }
    else
    {
        printf("Enter x");
        scanf("%lf",&geom1.object2D.x);
        printf("Enter y");
        scanf("%lf",&geom1.object2D.y);
    }
    printf("%lf", geom1.object2D.x);
    printf("%lf",geom1.object2D.y);

    geom1.oznaka = 1;
    if(geom1.oznaka)
    {
        printf("Enter x");
        scanf("%lf",&geom1.object3D.x);
        printf("Enter y");
        scanf("%lf",&geom1.object3D.y);
        printf("Enter z");
        scanf("%lf",&geom1.object3D.z);
    }
    else
    {
        printf("Enter x");
        scanf("%lf",&geom1.object2D.x);
        printf("Enter y");
        scanf("%lf",&geom1.object2D.y);
    }
    printf("%lf", geom1.object2D.x);
    printf("%lf",geom1.object2D.y);

    _getch();
    return 0;
}
```

Для змінних з типом об'єднання використовують ті ж операції, що й до структурних змінних: присвоєння, визначення адреси змінної, виділення окремих елементів (полів). Для звертання до елементів об'єднання використовують синтаксичні конструкції на основі операцій ".", "->", як і у випадку звертання до елементів структур:

Ім'я змінної об'єднання. Ім'я поля
Вказівник на об'єднання-> Ім'я поля
*(*Вказівник на об'єднання). Ім'я поля*

Наприклад:

```
union my
{
int grad;
double rad;
} My1;
```

```
My1.grad = 30;
printf("%d\n",My1.grad);
My1.rad = 3.14/6;
printf("%lf\n",My1.rad);
printf("%d\n",My1.grad);
```

Компілятор сам розпізнає, що елементи об'єднання зберігаються в одній і тій самій області пам'яті. У кожен момент часу може існувати лише один елемент – або `grad`, або `rad`. Щоб проілюструвати це, спочатку елементу `grad` присвоюємо значення 30 та виводимо це значення на екран. Тоді елементу `rad` присвоюємо значення 3.14/6 (π/6) та виводимо це значення на екран. Зрештою повторно виводимо на екран значення елемента `grad`. Воно вже не буде рівне 30, оскільки ця ж область пам'яті зайнята значенням `rad`.

2.5. Зчитування даних з текстового файлу

Масив структур зручніше заповнювати із попередньо заповненого текстового файлу, оскільки ввід даних з клавіатури з помилкою вимагатиме повторного набору всіх даних.

Часто буває потрібно прочитати весь файл. Для цього організовується цикл while, умовою продовження якого є те, що ще не досягнуто кінець файлу.

```
FILE *fp;
int symb;
//відкриття файлу
while ((symb=getc(fp))!=EOF)
{
// Оператори
}
```

Більшість функцій файлового обміну даними повертають значення, яке дає змогу перевірити, чи дана операція пройшла успішно, чи відбувся збій. Встановити конкретну причину збою можна через функції аналізу помилок файлового введення/виведення. Зокрема, функція

```
int feof(FILE*fp)
```

перевіряє, чи досягнуто кінець файлу, пов'язаного з потоком `fp`. Повертає 0, якщо не встановлено ознаку кінця файлу, інакше – ненульове значення. Всі операції читання з файлу після досягнення його кінця вважаються помилковими.

Функція

```
int ferorr(FILE*fp)
```

перевіряє, чи встановлено ознаку помилки в попередніх операціях звертання до потоків даних. Повертає нуль, якщо помилку незафіксовано, та ненульове значення, якщо виявлено помилку.

Причину збою в операціях потокового введення/виведення можна розшифрувати, використовуючи функцію

```
void perorr(char*errtext)
```

Функція виводить у стандартний потік помилок stderr текст повідомлення, записаний в `errtext ,` а за ним після двокрапки стандартне системне пояснення причини виникнення помилки.

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

```
int fclose(FILE*fp)
```

Функція має один параметр – вказівник на потік, який необхідно закрити. У разі успішного виконання вона повертає значення 0 .

3. КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке структура? Який загальний вигляд опису структури?
2. Які є підходи до опису структурної змінної?
3. Як здійснюється доступ до окремих полів структурної змінної?
4. Що таке вкладені структури? Як здійснюється доступ до полів вкладених структур?
5. Що таке об'єднання? Який загальний вигляд опису об'єднання?
6. Яка принципова різниця між структурою та об'єднанням?
7. Як здійснюється зчитування даних з текстового файлу?

4. ЛАБОРАТОРНЕ ЗАВДАННЯ

1. Ознайомитися з теоретичним матеріалом викладеним вище в даній інструкції і виконати приклади програм.
2. Одержати індивідуальне завдання.
3. Скласти програму на мові С у відповідності з розробленим алгоритмом.
4. Виконати обчислення по програмі.
5. Підготувати та здати звіт про виконання лабораторної роботи.

5. СПИСОК ЛІТЕРАТУРИ

1. Керниган Б., Ритчи Д. Язык программирования С. - М. - Финансы и статистика. - 1992. – 272 с.
2. Уэйт М., Прата С., Мартин Д. Язык С. Руководство для начинающих. - М. - Мир. - 1988. – 512 с.
3. К. Джамса. Учимся программировать на языке С++. М.: Мир, 1997. – 320 с
4. Герберт Шилдт. Полный справочник по С++. М. – С.-П.-К., Вильямс. – 2003. – 800 с.
5. Демидович Е. М. Основы алгоритмизации и программирования. Язык Си. (Учебное пособие). – Санкт-Петербург: "БХВ Петербург". – 2006. – 439 с.

6. ІНДИВІДУАЛЬНІ 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.

1. Відсортувавати дані за прізвищами студентів в алфавітному порядку. Визначити двох студентів з найвищим середнім балом.
2. Відсортувавати дані за прізвищами студентів в порядку протилежному алфавітному. Визначити п'ять студентів з найнижчим середнім рейтинговим балом.
3. Відсортувавати дані за віком студентів у зростаючому порядку. Роздрукувати список студентів з рейтинговим балом нижчим від середнього балу в групі.
4. Відсортувавати дані за віком студентів у спадному порядку. Роздрукувати список студентів з рейтинговим балом вищим від середнього балу в групі.
5. Роздрукувати список студентів, які отримали оцінки 2 на іспитах, у алфавітному порядку за прізвищем.
6. Роздрукувати список студентів, які отримали лише оцінки 5 на іспитах, у зростаючому порядку за віком.
7. Роздрукувати список студентів у зростаючому порядку за рейтинговим балом.
8. Роздрукувати список студентів, молодших середнього віку у групі, впорядкований у алфавітному порядку за прізвищем.

9. Роздрукувати список студентів, старших середнього віку у групі, впорядкований за зростанням рейтингового балу.
10. Роздрукувати список студентів, які отримали оцінки 4 і 5 на іспитах, у спадному порядку за віком. Визначити двох наймолодших студентів серед них.
11. Роздрукувати список студентів, які не отримали жодної оцінки 2 на іспитах, у алфавітному порядку за прізвищем. Визначити двох найстарших студентів серед них.
12. Роздрукувати список студентів, які не отримали жодної оцінки 5 на іспитах, впорядкований за середнім рейтинговим балом. Визначити найстаршого та наймолодшого студентів серед них.
13. Роздрукувати список студентів, народжених восени, впорядкований в алфавітному порядку за прізвищем.
14. Роздрукувати список студентів, народжених влітку, впорядкований в порядку зростання рейтингового балу.

З текстового файлу зчитати послідовність записів, які містять дані про книгу : <Автор>, <Назва книги>, <Рік видання>, <Кількість сторінок>, <Вартість>. Роздрукувати введені дані у вигляді таблиці, а також подати інформацію згідно варіанту.

15. Відсортувавати дані за прізвищами авторів в алфавітному порядку. Визначити дві книги з найбільшою кількістю сторінок.
16. Відсортувавати дані за прізвищами авторів в порядку протилежному алфавітному. Визначити п’ять найновіших книг за роком видання .
17. Відсортувавати дані за назвою в порядку протилежному алфавітному. Визначити шість найстаріших книг за роком видання .
18. Відсортувавати дані за назвою в алфавітному порядку. Визначити три найдорожчяих книги за вартістю.
19. Відсортувавати у зростаючому порядку дані за роком видання. Визначити три книги з найменшою кількістю сторінок.
20. Відсортувавати у зростаючому порядку дані за вартістю. Визначити книги авторів з прізвищем, що починається на букву А.
21. Відсортувавати у спадному порядку дані за роком видання. Визначити книги з назвою, що починається на букву А.
22. Відсортувавати у зростаючому порядку дані за кількістю сторінок. Визначити книги з кількістю сторінок, більшою за середню в бібліотеці.
23. Відсортувавати у спадному порядку дані за вартістю. Визначити книги видані пізніше 1980 року.
24. Відсортувавати у спадному порядку за вартістю дані про книги видані раніше 1975 року.
25. Відсортувавати за назвою в алфавітному порядку дані про книги, вартість яких більша середньої в бібліотеці.
26. Відсортувавати за прізвищем в алфавітному порядку дані про книги, вартість яких менша середньої в бібліотеці.
27. Відсортувавати у зростаючому порядку за роком видання дані про книги з кількістю сторінок меншою за середню в бібліотеці.