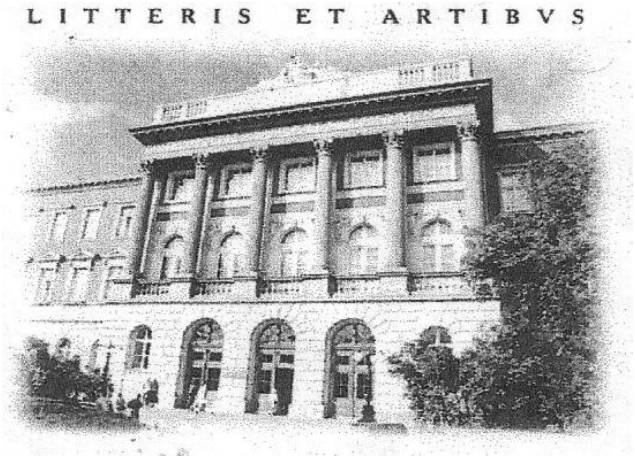
Міністерство освіти і науки України
Національний університет «Львівська політехніка»



Робота із структурами даних та структурованими файлами у програмах на С.

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи № 11
з дисципліни «Основи інформатики та програмування»
для студентів спеціальності 105 – "Прикладна фізика та наноматеріали"
для першого (бакалаврського) рівня освіти.

Затверджено
на засіданні кафедри
обчислювальної математики
та програмування
Протокол № 10 від 07.06.2018р.

Львів – 2018

Робота із структурами даних та структурованими файлами у програмах на С. Методичні вказівки до виконання лабораторної роботи № 11 для студентів спеціальності 105 – "Прикладна фізика та наноматеріали" для першого (бакалаврського) рівня освіти / Укл.: Гнатів Л.Б., Ментинський С.М., Пелех Я.М., Угрин С.З., 2018. - 20с.

Реєстр. № 8186 від 19.06.2018

Укладачі: Гнатів Л.Б., к. ф-м. н., доцент,
Ментинський С.М., ст. викл,
Пелех Я.М., к. ф-м. н., доцент,
Угрин С.З., асистент.

Відповідальний за випуск: Угрин С.З.

Рецензенти: Будз І.С., к.ф-м.н., доцент кафедри ОМП,
Філь Б.М., к.ф-м.н., доцент кафедри ОМП.

Передмова

У методичних вказівках розглянуто теоретичні та практичні аспекти групування різнотипних даних в структури та основні прийоми роботи зі структурованими файлами засобами мов програмування С та С++.

Методичні вказівки містять коротку довідкову інформацію, варіанти індивідуальних завдань та приклад розв’язування завдання одного з варіантів з поясненнями.

Методичні вказівки призначені для студентів спеціальності 105 – "Прикладна фізика та наноматеріали" для першого (бакалаврського) рівня освіти і укладені відповідно до робочої навчальної програми з дисципліни «Основи інформатики та програмування».

Лабораторні роботи №11

Робота із структурами даних та структурованими файлами у программах на С.

Мета роботи: Ознайомлення з структурованим типом даних struct та його використанням до конструювання нових типів даних шляхом об'єднання різнотипної інформації. Отримання практичних навичок опрацювання структурованих файлів з довільним доступом.

Короткі теоретичні відомості.

Поняття структури в С/С++.

Структури в програмуванні маєть багато спільного з усім відомими рядками таблиць. Те, що прийнято називати "шапкою" таблиці в мовах програмування носить назву шаблону структури. Наприклад, шаблон структури, яка описує дані про книгу, може бути таким:

```
struct book {
    char title[40];              //найменуванння
    char authors[30];            //автори
    char publishing_house[15];   //видавництво
    int year;                    //рік видання
    int pages;                   //кількість сторінок
    float price;                 //ціна
  }
```

Далі ***book*** можна використовувати для оголошення конкретних змінних:

```
struct book b1,b2,b3;   //змінні типу book
```

В мові С++ при оголошенні змінних службове слово ***struct*** можна опустити:

```
book b1,b2,b3;
```

Оголошення шаблону структури і змінних, пов'язаних з цією структурою, можна поєднати:

```
struct book {
    char title[40];
    char authors[30];
    char publishing_house[15];
    int year;
    int pages;
    float price;
  } b1,b2,b3;
```

Або без назви типу:

```
struct {
    char title[40];
    char authors[30];
    char publishing_house[15];
    int year;
    int pages;
    float price;
  } b1,b2,b3;
```

Проте для передачі параметрів у функції таке задання непридатне.

Ініціалізація та доступ до полів структур

Змінна типу структура може бути ініціалізована, як і змінна будь-якого стандартного типу, тобто при оголошенні можна надати значення цій змінній. Для цього після імені (ідентифікатора) оголошуваної змінної через знак дорівнює («=») у фігурних дужках через кому надається початкове значення кожному полю структури, враховуючи заданий при оголошенні порядок полів структури. Слід дотримуватися відповідності типів полів та типів їх значень. Наприклад:

```
      struct detal
{     int nomer_modeli, nomer_detali;
      float vartist_detali;
};
```

Тоді змінні типу detal можна оголошувати так:

```
detal det, det1={12089, 245, 187.57f}, det2;
```

Доступ до полів структурної змінної здійснюється через операцію крапка (.). Для того щоб отримати доступ до відповідного поля структурної змінної використовують наступну синтаксичну конструкцію:

```
ім'я_змінної.ім'я_поля
```

Така конструкція може входити у вираз, якщо поле вже містить дані, бути аргументом функції, бути лівою частиною оператора присвоєння, якщо це поле не оголошено константним. Над кожним з полів структурної змінної можуть виконуватися операції, які визначені для відповідного типу даних.

Наприклад, щоб вивести вміст структурованої змінної на екран, необхідно виводити кожне поле зокрема за інструкцією наступного вигляду

```
cout<<"\nНомер моделі : "<<det1.nomer_modeli;
cout<<"\nНомер деталі :" <<det1.nomer_detali;
cout<<"\nВартість деталі ="<<det1.vartist_detali<<" грн.\n";
```

Операція доступу до поля структури використовується також при введенні значень полів структури, при чому кожного поля зокрема. Нехай треба ввести нову інформацію про деталь. Використовуючи інструкції виводу та вводу, заповнимо відповідні поля структурної змінної

```
cout<<"\nВведіть дані про деталь det:\n";
cout<<"\n Номер моделі ="; cin>>det.nomer_modeli;
cout<<"\n Номер деталі ="; cin>>det.nomer_detali;
cout<<"\n Вартість деталі ="; cin>>det.vartist_detali;
```

Якщо оголошено вказівник на змінну типу структура, тоді доступ до відповідного поля здійснюється поєднанням операції крапка з операцією розадресації. Та оскільки операція взяття поля структури має вищий пріорітет від операції розадресації, тоді для взяття поля за адресою спершу треба звернутися за цією адресою, а тоді доступатися до поля. Це означає, що операцію розадресації слід взяти в дужки, підвищивши її пріорітет:

```
(*вказівник_наструктуру).ім'я_поля_структури.
```

У мовах C/C++ операція непрямого доступу до поля, тобто взяття вмістимого поля за адресою, що у вказівникові, подається символьним зображенням операції стрілка вліво(->). Отже цю операцію записують двома послідовними символами: мінус, більше('-','>'):

```
detal * pd=&det1;
pd->nomer_detali=25;
```

Таким чином, можемо встановлювати адресу структури, за цією адресою використовувати вмістиме з допомогою операцій -> або (*).

Ключове слово typedef та його застосування до структур

У мові С як при створенні структур так і при оголошенні змінних необхідно перед іменем користувацького типу додавати ключове слово ***struct***, що спричинює певну незручність. Граматика мови С передбачає можливість створення власний назв як стандартних, так і анрегованих типів даних з використанням ключового слова ***typedef***. Це ключове слово дозволяє скоротити описові інструкції, а також надати нове псевдо відомому типові. Наприклад:

```
unsigned long long a;
typedef unsigned long long UnLL;
UnLL b, c;
```

Ідентифікатор *UnLL* типу може тепер використовуватися як параметр у функціях у всій області його видимості.

Нехай ми опрацьовуємо матриці, тоді можна створити назву типу, що відповідатиме за статичний двовимірний масив. При цьому можна задавати тип елемента масиву, а також максимально можливу розмірність. Задане нове ім'я використовується у оголошенні формального параметра, оголошенні змінних,

що їх компілятор сприйматиме як даовимірний масив з відповідним виділенням пам’яті.

```
typedef int chuslo;
const int rd=20, sc=25;
typedef chuslo matrix[rd][sc];
void vvid_mtr(matrix a,int n,int m,char*ms);
matrix a1,a2;
```

Ключову слово typedef дозволяє спростити текст програми та зекономити зусилля зокрема при оглошенні структурних змінних. Для спрощення подальшого оголошення структурний змінних використовується оператор typedef. Розглянемо застосування для оголошення структури, що задає комплексне число

```
typedef struct COMPLEX
{double Re,Im;}complex;
complex cc={1,2};
```

При цьому можна і не вказувати імені структури, а просто вказувати нове ім’я оголошуваного типу даних:

```
typedef struct
{int chus, znam; } drib;
typedef struct
{double x, y, z; } Point_3D;
```

. Використання структур

Для структур, оголошених з використанням одного і того ж шаблону, допустима операція присвоювання:

b1=a; //все поля структуры a копируются в структуру b1

На жаль, однойменні поля строкового типу у структур так копіювати не можна - необхідно вдаватися до послуг функцій типу strcpy:

strcpy(b1.authors,a.authors); //копируем поле "авторы"

Структури можуть бути аргументами функцій. Якщо функція не змінює структуру, то таку структуру можна передати за значенням. Якщо обробка структури в функції пов'язана із зміною вмісту полів, то таку структуру необхідно передавати за вказівником або за посиланням.

У деяких таблицях використовуються багатоповерхові шапки, коли певне поле заголовка, у свою чергу, розпадається на кілька полів. Така ж ситуація може зустрітися і в структурах, коли в якості чергового поля виступає інша структура. І рівень вкладення таких полів нічим не обмежений.

Функції можуть не тільки одержувати структури в якості своїх параметрів, але і повертати результати у вигляді структур. І це означає, що функція, що

повертає значення може мати в якості результату своєї роботи сукупність значень полів відповідної структури.

Опрацювання файлів у C/C++

В більшості своїй файли являють собою іменовані області зовнішньої (дискової) пам'яті, з якими програми можуть обмінюватися інформацією. Необхідність у таких обмінах, по-перше, виникає, коли обсяг оперативної пам'яті недостатній для зберігання потрібної інформації. По-друге, програма може скористатися даними, отриманими раніше іншою програмою і завбачливо записаними на диск. Нарешті, в програмах, які вимагають під час своєї роботи введення вихідних даних досить великого обсягу, доцільно зчитувати ці дані з файлу - дані у файлі можна підготувати завчасно і ретельно вивірити. До числа абонентів, які можуть брати участь в обміні даними, відносяться і файли-пристрої (дисплей, принтер, плотер, сканер, клавіатура, канали зв'язку і т.п.). Даний розділ присвячений роботі з дисковими файлами, хоча технологія обслуговування файлів-пристроїв нічим принципово не відрізняється.

При використанні імен файлів в програмах на мовах С, С + + слід пам'ятати про те, що специфікація файлу задається рядком, в якій знак "зворотного слеша" замінюється подвоєним зворотним слешем:

```
char namef[]="c:\\bc\\bin\\bc.exe"
```

Оцінюючи ключові аспекти процесу обміну даними, можна сказати, що робота з файлами, в основному, обмежується **трьома-чотирма операціями**:

виділення ресурсів та ***приведення файлу в стан готовності до обміну*** (саме це приховується за терміном "відкрити файл");

читання (введення з файлу) або **запис** (висновок у файл) чергової порції даних;

повернення виділених ресурсів і ***завершення незакінчених операцій*** (цьому відповідає термін "закрити файл").

Використовують два типии файлів: файли ***послідовного*** доступу і файли ***довільного*** доступу

Текстові файли

Текстові файли належать до файлів послідовного доступу, оскільки одиницею зберігання інформації в них є рядки змінної довжини. Кожен рядок закінчується спеціальним символом ‘\0’ (нуль-байт). Найважливішою перевагою текстових файлів є універсальність формату зберігання інформації - числові дані в символьному вигляді доступні на будь-якому комп'ютері, при необхідності їх може прочитати і людина. Однак ця перевага має і зворотний бік медалі - перетворення числових даних з машинних форматів в символьний вигляд при виведенні і зворотне перетворення при введенні пов'язане з додатковими витратами. Крім того, обсяг числових даних в символьному

форматі займає в кілька разів більше пам'яті в порівнянні з їх машинним представленням.

Для ініціалізації текстового файлу необхідно завести вказівник на структуру типу FILE та відкрити файл по оператору fopen в одному з потрібних режимів - "rt" (текстовий для читання), "wt" (текстовий для запису), "at" (текстовий для дозапису в уже існуючий набір даних):

```
FILE *f1;

  f1=fopen(імя_файлу, "режим");
```

Формат оператора обміну з текстовими файлами мало чим відрізняється від операторів форматного введення (scanf) і виводу (printf). Замість них при роботі з файлами використовуються функції fscanf і fprintf, у яких єдиним додатковим аргументом є вказівник на відповідний файл:

```
fscanf(f1,"список_форматів", список_вводу);

  fprintf(f1,"список_форматів \n",список_виводу);
```

Якщо чергова рядок текстового файлу формується із значення елементів символьного масиву str, то замість функції fprintf простіше скористатися функцією ***fputs (f1, str).*** Читання повної рядки з текстового файлу зручніше виконати за допомогою функції ***fgets (str, n, f1)***. Тут параметр n означає максимальну кількість прочитуваних символів, якщо раніше не зустрінеться керуючий байт 0A.

Двійкові файли

Двійкові файли відрізняються від текстових тим, що являють собою послідовність байтів, вміст яких може мати різну природу. Створення двійкових файлів за допомогою функції fopen відрізняється від створення текстових файлів тільки зазначенням режиму обміну - "rb" (двійковий для читання), "rb +" (двійковий для читання і запису), "wb" (двійковий для запису), "wb +" (двійковий для запису і читання):

```
FILE *f1;

  f1=fopen(имя_файла, "режим");
```

Зазвичай для роботи з бінарними файлами використовують функції fread и fwrite:

```
c_w = fwrite(buf, size_rec, n_rec, f1);
```

де

buf – вказівни типу void* на початок буфера в оперативній пам’яті, де міститься інформація, що буде записана (куди зчитується) у файл;

size_rec – размір порції в байтах;

n_rec – к-сть порцій, які записують у файл;

f1 – вказівник на файл;

c_w – к-сть порцій, які були фактично записані в файл.

Функція **fread** з тими ж параметрами використовується для запису в бінарний файл:

```
c_r = fread(buf, size_rec, n_rec, f1);
```

тут **c_r** – к-сть порцій, прочитаних з файла.

Двійкові файли допускають не тільки послідовний обмін даними. Контроль за поточною позицією доступних даних у файлі здійснює система за допомогою вказівника, що знаходиться в блоці управління файлом. За допомогою функції ***fseek*** програміст має можливість перемістити цей вказівник:

```
fseek (f1, delta, pos);
```

тут

delta - величина зміщення в байтах, на яку слід перемістити вказівник файлу;

pos - позиція, від якої проводиться зсув вказівника (**0** або **SEEK_SET** - від початку файлу, **1** або **SEEK_CUR** - від поточної позиції, **2** або **SEEK_END** - від кінця файлу)

Структуровані файли

Структурований файл є окремим випадком двійкового файлу, в якому в якості порції обміну виступає структура мови С, що є точним аналогом записи в Паскалі. У порівнянні з попереднім прикладом використання записів дозволяє скоротити кількість звернень до функцій fread / fwrite, тому в одному зверненні беруть участь всі поля запису.

Ініціалізація структурованого файла виконується точно таким же способом, як і підготовка до роботи двійкового файлу. Наприклад:

```
 FILE *f1;
  int j,k;
  struct {    char s[5];        int n;       float r;    } b;
  f1=fopen("c_rec","wb");
      fwrite(&b,sizeof(b),1,f1);
  fclose(f1);
  f1=fopen("c_rec","rb");
      fread(&b,sizeof(b),1,f1);
```

Завдання 1. Описати структуру даних (struct) згідно варіанта. Скласти програму, яка читає значення полів структури з консолі та прочитану структуру у бінарний файл. Передбачити до запис даних у файл так, щоб внаслідок виконання програми файл містив 10-15 екземплярів структурованих даних. Після завершення вводу даних вивести увесь вміст файла на консоль у вигляді таблиці.

Варіанти завдань

1. Створити файл, який містить дані про авіарейси. Запис містить поля: номер рейсу (рядкова змінна), пункт призначення, час відправлення (рядкова змінна), ціна білета, кількість вільних місць.

2. Створити файл, який містить дані про кількість виробів категорій A, B, C, які зібрані робітниками заводу за місяць. Запис містить поля: прізвище робітника, номер цеху, кількість зібраних за місяць виробів категорії A, кількість зібраних за місяць виробів категорії B, кількість зібраних за місяць виробів категорії C.

3. Створити файл, який містить дані про здачу сесії. Запис містить поля: шифр групи, прізвище студента, назва предмета, оцінка, назва предмета...(всього 5 предметів).

4. Створити файл, який містить дані про продаж комп'ютерної техніки. Запис містить поля: код покупця, назва товару, ціна за одиницю товару, продана кількість.

5. Створити файл, який містить дані про рух пасажирських поїздів від ст. Львів. Запис містить поля: назва кінцевої станції, номер поїзда, час відправлення , наявність місць по категоріях вагонів- СВ, купейний, плацкартний, загальний.

6. Створити файл, який містить дані про книжковий фонд бібліотеки. Запис містить поля: шифр книги, назва, прізвище автора, рік видання.

7. Створити файл, який містить дані про курси обміну валют в банках м. Львова. Запис містить поля: назва банка, назва валюти, курс купівлі, курс продажу, назва валюти,...Назви валют: американський долар, марка, фунт, злотий.

8. Створити файл, який містить дані про міжміські автобусні рейси. Запис містить поля: назва міста призначення, час відправлення, ціна білета.

9. Створити файл, який містить дані про автомобілі та їх власників. Запис містить поля: номер автомобіля, марка, колір, рік випуску, прізвище власника.

10. Створити файл, який містить дані про студентів, які мешкають в гуртожитку №.. Запис містить поля: прізвище студента, номер кімнати, назва факультету, шифр групи.

11. Створити файл, який містить дані про сплату за електроенергію мешканцями будинку №. Запис містить поля: прізвище господаря, номер квартири, останнє сплачене значення по лічильнику, поточне значення лічильника.

12. Створити файл, який містить дані про оплату послуг міжміського телефонного зв'язку. Запис містить поля: прізвище володаря, номер телефону, назва міста, дата, тривалість, сума.

13. Створити файл, який містить дані про туристичні маршрути. Кожен запис містить дані: назва тура, ціна путівки, назва міста, кількість днів перебування в місті, назва міста, кількість днів перебування в місті,...(5 пунктів на маршруті).

14. Створити файл - телефонний довідник міста. Кожен запис містить поля: прізвище, ініціали, номер телефону, назва вулиці, номер будинку.

15. Створити файл - довідник про дні народження рідних та друзів. Кожен запис містить поля: ім'я, число, місяць, рік народження.

16. Створити файл, який містить дані про книги в книжковому магазині. Запис містить поля: прізвище автора, назва, рік видання, кількість сторінок, ціна.

Приклад.

```
#include<iostream>
using namespace std;
#pragma warning(disable:4996)
struct book{
	char author[50];
	char title[200];
	int year, pages;
	double price;
};
book inputBooks(book b)
//функція читання даних про чергову книгу
{
	setlocale(LC_ALL, "ukr");
	cout<<"Введіть дані про книгу\n Автор:\n";
	setlocale(LC_ALL, "C");
	cin.getline(b.author,50);
	setlocale(LC_ALL, "ukr");
	cout<<"Назва:\n";
```

```
	setlocale(LC_ALL, "C");
	cin.getline(b.title,200);
	setlocale(LC_ALL, "ukr");
	cout<<"Рік видання"<<endl;	cin>>b.year;
	cout<<"Кількість сторінок:"<<endl; cin>>b.pages;
	cout<<"Ціна:"<<endl;	cin>>b.price;
	cin.get();
	return b;
}
void main()
{
	book myBook = {"","",0};
	FILE *f;
	if(f = fopen("E:/books.dat","a+b"))
		{
		char again;
		do{
			myBook = inputBooks(myBook);
			fwrite(&myBook, sizeof(myBook),1, f);
			system("cls");
			setlocale(LC_ALL, "ukr");
			cout<<"Ще одну книгу? (y/n)";
			cin>>again;
		}while(again!='n'&& again!='N');
		fclose(f);
		//друк даних зі створеного файла
		cout<<"Список книг:\n";
		f = fopen("E:/books.dat","rb");
		setlocale(LC_ALL, "C");
		while(true)
		{
			fread(&myBook, sizeof(book),1,f);
			if(feof(f)) break;
			cout<<myBook.title<<" // "<<myBook.author;
			cout<<". "<<myBook.year<<"p. - "<<myBook.pages;
			cout<<" p.,\tPrice:\t  "<<myBook.price<<" uah. "<<endl;

		}
		fclose(f);
	}
	else
	{
		setlocale(LC_ALL, "ukr");
		cout<<"Error: файл не створено";
	}
}
```

Завдання 2. Скласти програму, яка опрацьовує файл, утворений програмою з завдання 1. Програма повинна виводити у вигляді таблиці увесь вміст файла, далі дати користувачеві змогу вибрати критерій пошуку даних за одним з полів структури у вигляді текстового меню з 3-4 пунктів. Наприклад, для структури "Книга" можна створити меню:
1 – пошук книг за автором
2. – пошук книг за частиною назви
3 – пошук книг за роком видання
4. – завершити роботу з файлом.
Після вибору користувачем відповідного пункту меню, програма повинна запитувати і зчитувати критерій для пошуку, та виводити у вигляді таблиці лише ті дані з файла, які відповідають заданому критерію
До меню обов'язково включити пункт пошуку, що сформульовано у варіантах завдання нижче.

Варіанти завдань

1. По введених з клавіатури назві пункту призначення вивести у виді таблиці інформацію про всі рейси літаків в заданий пункт, а саме: номер рейсу, час відправлення, ціну білета.

2. Ввести з клавіатури вартість зборки виробів по категоріям (окремо для виробів А, виробів В, виробів С). Вивести інформацію про заробіток робітника , прізвище якого ввести з клавіатури.

3. Вивести прізвища студентів, які отримали "5" з усіх предметів.

4. По введеному з клавіатури коду покупця вивести список куплених ним товарів і загальну суму, на яку покупець купив товар.

5. По введеній з клавіатури назві кінцевої станції та поточний час вивести інформацію про поїзди, які відправляються в заданому напрямку та наявність місць по категоріям вагонів (дані вивести в табличному виді).

6. По введеній з клавіатури назві книги та прізвищу автора вивести повну інформацію про книгу (шифр книги, назва, прізвище автора , рік видання).

7. По назві валюти, введеній з клавіатури, вивести назву банка, в якому курс купівлі валюти найбільший.

8. По введеній з клавіатури назві міста призначення вивести в табличному виді інформацію про рейси в даному напрямку: час відправлення, ціна білета.

9. По введеному з клавіатури року випуску автомобілів вивести повну інформацію про них: номер автомобіля, марка, колір, прізвище власника.

10. По введеному з клавіатури шифру групи вивести в табличному виді повну інформацію про студентів групи, яки проживають в гуртожитку : прізвище студента, номер кімнати, назва факультету.

11. По введеному з клавіатури номери квартири вивести прізвище власника та суму, яку потрібно сплатити за електроенергію.

12. По введеному з клавіатури номеру телефону вивести назву міста , куди дзвонили і коли.

13. По введеній з клавіатури назві тура роздрукувати ціну путівки, порахувати та роздрукувати вартість одного дня відпочинку на даному маршруті.

14. По введеному з клавіатури номеру АТС (перші дві цифри телефонного номера) вивести в табличному виді дані про абонентів (прізвище, ініціали, номер телефону, назва вулиці, номер будинку).

15. По введеній з клавіатури назві поточного місяця роздрукувати дані про дні народження в поточному місяці. Якщо дата «кругла» (кількість років ділиться на 5) вивести примітку - текстове повідомлення «Ювілей».

Приклад

```
#include<iostream>
using namespace std;
#pragma warning(disable:4996)
struct book{
	char author[50];
	char title[200];
	int year, pages;
	double price;
};
void printByAuthor(char a[], FILE *p);
void printByTitle(char t[], FILE *p);
void printByYear(int y, FILE *p);


void main()
{
	FILE *f;
	book myBook;
	if(f = fopen("E:/books.dat","r"))
		{
```

```
setlocale(LC_ALL, "ukr");
system("cls");
cout<<"Список книг:\n";
f = fopen("E:/books.dat","rb");
setlocale(LC_ALL, "C");
int m = fread(&myBook, sizeof(book),1,f);
while( !feof(f))
{
      cout<<myBook.title<<" // "<<myBook.author;
      cout<<". "<<myBook.year<<"p. - "<<myBook.pages;
      cout<<" p.,\tPrice:\t  "<<myBook.price<<" uah. "<<endl;
      int m = fread(&myBook, sizeof(book),1,f);
}
fclose(f);
cout<<endl<<endl;
int pointM, fyear;
char fauthor[50], parttitle[200];

while(true)//меню виводиться доки користувач не обере вихід
{
setlocale(LC_ALL, "ukr");
cout<<"Оберіть дію \n1 - пошук книг за автором\n";
cout<<"2 - пошук книг за назвою\n";
cout<<"3 - пошук книг за роком видання\n";
cout<<"4 - завершити роботу з програмою\n";
cin>>pointM;
f = fopen("E:/books.dat","rb");
switch(pointM)
{
case 1: cout<<"Прізвище автора: \n";
           setlocale(LC_ALL, "C");
           cin.get();
           cin.getline(fauthor, 50);
           printByAuthor(fauthor, f);
           fclose(f);
           break;
case 2: cout<<"Назва книги, або її частина: \n";
           setlocale(LC_ALL, "C");
           cin.get();
           cin.getline(parttitle, 200);
           printByTitle(parttitle, f);
           fclose(f);
           break;
case 3: cout<<"Рік видання: \n";
           cin>>fyear;
           printByYear(fyear, f);
           fclose(f);
           break;
case 4: fclose(f);
           return;
}
```

```
		}
	}
	else
	{
		setlocale(LC_ALL, "ukr");
		cout<<"Error: файл не знайдено";
	}
}

void printByYear(int y, FILE *p)
{
		setlocale(LC_ALL, "ukr");
		cout<<"Список знайдених книг:\n";
		book b;
		setlocale(LC_ALL, "C");
		fread(&b, sizeof(book),1,p);
		while( !feof(p))
		{
			if(y == b.year)
			{
				cout<<b.title<<" // "<<b.author;
				cout<<". "<<b.year<<"p. - "<<b.pages;
				cout<<" p.,\tPrice:\t  "<<b.price<<" uah. "<<endl;
			}
			fread(&b, sizeof(book),1,p);
		}
}

void printByAuthor(char a[], FILE *p)
{
		setlocale(LC_ALL, "ukr");
		cout<<"Список знайдених книг:\n";
		book b;
		setlocale(LC_ALL, "C");
		fread(&b, sizeof(book),1,p);
		while( !feof(p))
		{
			if(!strcmp(a, b.author))
			{
				cout<<b.title<<" // "<<b.author;
				cout<<". "<<b.year<<"p. - "<<b.pages;
				cout<<" p.,\tPrice:\t  "<<b.price<<" uah. "<<endl;
			}
			fread(&b, sizeof(book),1,p);
		}
}

void printByTitle(char t[], FILE *p)
	{
		setlocale(LC_ALL, "ukr");
		cout<<"Список знайдених книг:\n";
		book b;
```

```
            setlocale(LC_ALL, "C");
            fread(&b, sizeof(book),1,p);
            while( !feof(p))
            {
                  if(strstr(b.title, t))
                  {
                        cout<<b.title<<" // "<<b.author;
                        cout<<". "<<b.year<<"p. - "<<b.pages;
                        cout<<" p.,\tPrice:\t  "<<b.price<<" uah. "<<endl;
                  }

                  fread(&b, sizeof(book),1,p);
            }
}
```

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Який синтаксис умовної конструкції в C/C++?
2. Який результат операцій відношення в мові C++?
3. Яке призначення оператора ***else***?
4. В чому полягає відмінність між повною та короткою формою розгалуження?
5. Що таке тернарний оператор (операція)?
6. Які дії виконує оператор switch ?
7. Яке призначення блоку default в операторі switch?

Рекомендована література:

1. Програмування на С(С++). Парадигма процедурного програмування. Навчальний посібник. /Гнатів Б.В., Гнатів Л.Б. . – Львів: Видавництво «Растр-7», 2017. – 264 с.
2. Основи програмування на С/С++. Конспект лекцій з курсу «Основи інформатики і програмування, частина 2» спеціальності 105 – «Прикладна фізика та наноматеріали» для першого (бакалаврського) рівня освіти/ Укл.: Ментинський С.М., 2016. – 140 с
3. П. Каленюк, І. Клюйник, І. Кравець, Л. Гнатів, Л.Демків, І. Подун Основи програмування в середовищі С. Лекції та завдання до лабораторних робіт. Львів 2007 р., 98 с.
4. Петрович Р. Й., Тумашова О. В. “Основи програмування мовою Сі.”. Навчальний посібник. Видавництво НУ”ЛП”, Львів, 2005р. 100с.
5. Програмування мовою С: навчальний посібник для вузів / Зореслава Ярославівна Шпак . — Львів: Оріяна-Нова, 2006 . — 431 с.
6. Кравець П.О. Об’єктно-орієнтоване програмування: навч.посібн. / П.О.Кравець.— Львів: Вид-во Львівської політехніки, 2012.— 624 с.
7. Креневич, А.П. С у задачах і прикладах : навчальний посібник із дисципліни "Інформатика та програмування" / А.П. Креневич, О.В. Обвінцев. – К. : Видавничо-поліграфічний центр "Київський університет", 2011. – 208 с.
8. Шилдт, Герберт. Полный справочник по C++, 4-е издание. . Пер. с англ. — М. : Издательский дом “Вильямс”, 2006. — 800 с. : ил.
9. Стивен Пратта Язык программирования С. Лекции и упражнения - К.: Диасофт, 2002.

НАВЧАЛЬНЕ ВИДАННЯ

Робота із структурами даних та структурованими файлами у програмах на С.

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи №11
з дисципліни «Основи інформатики та програмування»

для студентів спеціальності 105 – "Прикладна фізика та наноматеріали" для першого (бакалаврського) рівня освіти.

Укладачі: Гнатів Л.Б., к. ф-м. н., доцент,
Ментинський С.М., ст. викл,
Пелех Я.М., к. ф-м. н., доцент,
Угрин С.З., асистент.

Комп’ютерне складання: Ментинський С.М., ст. викл.