

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет "Запорізька політехніка"

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт із дисципліни

"Програмування"

для студентів спеціальності 123 «Комп'ютерна інженерія»
скороченої форми навчання

Робота з файлами

Методичні вказівки до виконання лабораторних робіт із дисципліни "Програмування" для студентів спеціальності 123 "Комп'ютерна інженерія" скороченої форми навчання. Робота з файлами / Укл.: Н.В. Луценко. – Запоріжжя: НУ "Запорізька політехніка", 2021. – 18 с.

Укладачі:

Н.В. Луценко, ст. викладач

Рецензент:

Р.К.Кудерметов, к.т.н., доцент

Відповідальний за випуск

Н.В. Луценко, ст. викладач

Затверджено
на засіданні кафедри КСМ
Протокол № 6 від 05.02.2021

Затверджено
на засіданні НМК КНТ
Протокол № 7 від 26.02.2021

ЗМІСТ

Порядок виконання лабораторних робіт.....	4
Загальні відомості	4
Лабораторна робота. Робота з файлами.....	11
Програма 1.	11
Завдання до програми 1.....	12
Програма 2.	13
Завдання до програми 2.....	14
Програма 3.	15
Завдання до програми 3.....	16
Контрольні питання до лабораторної роботи	17
Рекомендована література.....	18

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Лабораторна робота складаються з декількох програм, завдань до них і контрольних питань. Студент повинен розробити, відпрацювати і зберегти у робочому каталозі програму, що наведена, й програму, розроблену згідно заданого варіанта, та проаналізувати результати їхньої роботи. За результатами виконання лабораторної роботи студент складає й захищає письмовий звіт, вимоги до якого викладені нижче.

Зміст звіту

1. Титульний лист, виконаний відповідно до СТП.
2. Мета роботи.
3. Відповіді на контрольні питання.

ЗАГАЛЬНІ ВІДОМОСТІ

Функції роботи з файлами. Для обміну даними з початком виконання програми автоматично відкриваються стандартні потоки: 1) введення даних з клавіатури (**stdin**, **cin**); 2) виведення даних на екран монітору (**stdout**, **cout**); 3) виведення даних на екран монітору повідомлень про помилки (**stderr**, **cerr**).

Усе інші потоки для обміну даними між файлами потрібно відкривати явно. Ці потоки можна відкривати в одному напрямку (тільки для читання або запису) та в двох напрямках (для читання та запису).

Функції обміну з файлами працюють в двох режимах: текстовому і двійковому. За замовченням встановлюється текстовий режим. В *текстовому режимі* при запису в файл символ '\n' перетворюється в два символи '\n' та '\0', а при читанні з файлу такі символи сприймаються як два символи '\n' и '\0'. При обміні з файлом в *двійковому режимі* ніяких перетворень символів не виконується.

Для роботи з файлами можна використовувати або структуру типу **FILE**, що визначено в файлі **<stdio.h>**, або об'єкти класів, що визначені в файлі **<fstream.h>**.

Визначення потоку структурного типу FILE. При роботі зі структурою типу **FILE** спочатку визначається показчик на цю структуру (потік), яка містить всю інформацію для роботи з потоком, потім відкривається потік для зв'язку з файлом. Наприклад:

```
FILE *fp; // визначення показчика на структуру типу FILE
```

Відкриття і закриття файлу. Для відкриття потоку для зв'язку з файлом використовується функція **fopen()**, що повертає показчик на структуру типу **FILE**, якщо файл успішно відкритий, або **NULL**-показчик, якщо файл не відкрився. Прототип функції має вигляд:

```
FILE *fopen(const char *filename, const char *mode);
```

де **filename** – показчик на рядок, що задає шлях та ім'я файлу, якщо шлях не задано, то вибирається поточний диск, директорій;

mode – показчик на рядок символів, що задають режим відкриття.

Деякі комбінації цих символів приведені нижче:

w – потік відкривається тільки для запису у файл, **w+** – потік відкривається для запису у файл і для читання з файлу. Якщо файлу немає, він створюється, якщо файл існує, його старий вміст пропадає;

r – потік відкривається для читання з існуючого файлу, **r+** – потік відкривається для читання і запису в існуючий файл;

a – потік відкривається для додавання (запису даних в кінець файлу),

a+ – потік відкривається для додавання в файл і для читання з файлу. Якщо файл не існує, він створюється;

t – потік відкривається в текстовому режимі, задається після **r, w, a**;

b – потік відкривається в двійковому режимі, задається після **r, w, a**.

Закриття файлу виконує функція **fclose()**, що має прототип:

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

Функція закриває потік, на який указує **fp**. При успішному закритті файлу функція повертає **0**, інакше – **EOF**. При цьому вилучається структура типу **FILE**, оновлюється зміст директорія (інформацію про розмірі, дату та час).

Наприклад:

```
FILE *fp=fopen("D:\\User\\file.txt", "w") ; // відкриття потоку  
if (fp==NULL) // перевірка відкриття файлу  
{ puts("помилка відкриття файлу"); return -1; }  
...  
fclose(fp) ; // закриття потоку
```

Функції запису в файл та читання з файлу. Нижче приводяться функції запису (введення) в файл та читання (виведення) з файлу, які найчастіше застосовуються.

Функція запису символів в файл

```
int fputc(int ch, FILE *fp); // прототип
```

Функція у випадку успіху записує у відкритий файл символ **ch** після його перетворення в тип **char**, інакше – **EOF**.

Наприклад:

```
char c1 = 'A'; // визначення та ініціалізація символу  
fputc(c1, fp); // запис символу в файл
```

Функція читання символів з файлу

```
int fgetc(FILE *fp); // прототип
```

Функція при успіху повертає прочитаний символ після його перетворення в **int** без врахування знаку. У випадку помилки чи досягнення кінця файлу повертається **EOF**.

Наприклад:

```
char c2; // резервування пам'яті під символ  
c2=fgetc(fp); // читання символу з файлу  
putch(c2); //виведення символу на екран
```

Функція запису рядка в файл

```
int *fputs(char *str, FILE *fp) ; // прототип
```

Функція переносить байти символів з рядка, на який вказує покажчик **str**. Перенесення закінчується при досягненні символу '**\0**' (кінець рядка), який в файл не записується..

Наприклад:

```
char str1[20]="Програмування"; //визначення та ініціалізація рядка  
fputs(str1, fp); // запис рядка в файл
```

Функція читання рядка з файлу

```
char *fgets(char *str, int N, FILE *fp); // прототип
```

Функція приймає байти з відкритого файлу і розміщує їх за адресом, що заданий покажчиком **str**. Прийом символів закінчується після передачі **N** байт чи при одержанні символу '**\n**'. У випадку успіху повертає покажчик на прочитаний рядок символів, інакше – **NULL**.

Наприклад:

```
char str[21]; // резервування пам'яті під рядок (21 байт)  
fgets(str, 21, fp); // читання рядка з файлу не більш 21 байтів  
puts(str); // виведення рядка на екран
```

Функція форматowanego запису даних в файл

```
int fprintf(FILE *fp, char *format[,arg]...); // прототип
Функція записує значення даних з перетворенням за вказаним
форматом у відкритий файл. Наприклад:
int d=7; float f=3.5; char g='B'; //визначення змінних
fprintf (fp,"%d %f %c",d,f,g); // запис змінних в файл
```

Функція форматowanego читання даних з файлу

```
int fscanf(FILE *fp, char *format[,pointer]...); // прототип
Функція зчитує з відкритого файлу значення даних з перетворенням за
вказаним форматом і розміщує їх за вказаними адресами. Наприклад:
int a; float b; char c; // резервування пам'яті під змінні
fscanf(fp,"%d %f %c",&a,&b,&c); //читання змінних з файлу
printf("%d %f %c\n",a,b,c); //виведення змінних на екран
```

Функція запису блоків даних в файл

```
int fwrite(void *ptr,int size,int N,FILE* fp); // прототип
Функція виконує запис даних у відкритий файл N блоків, розміром
size байт кожен. Запис виконується у двійковому режимі "wb".
struct REC{int a; float b[2];};
REC st1={1, 2.2, 3.3}; //визначення структури (блок з 10 байт)
fwrite(&st1,sizeof(REC),1,fp); //запис в файл структури
```

Функція читання блоків даних з файлу

```
int fread(void *ptr,int size,int N,FILE *fp); // прототип
Функція виконує читання з відкритого файлу N блоків, розміром size
байт кожен. Читання виконується у двійковому режимі "rb".
REC st3; // резервування пам'яті під блоки даних
fread(&st3,sizeof(REC),1,fp); //читання з файлу структури
// виведення на екран зчитаної структури
printf("%d %.2f %.2f\n",st3.a,st3.b[0],st3.b[1]);
```

Довільний доступ до файлу. При обміні даними з файлом (записі і читанні) місце, куди записується або зчитується інформація, визначається покажчиком запису-читання. У бібліотеці мови Сі є функції, що керують положенням покажчика запису-читання..

Функція установлення покажчика запису-читання в довільну точку:

```
int fseek(FILE *fp, long offset, int from_where);
Встановлюється покажчик запису-читання в потоці fp на offset байт
управо або вліво. Параметр from_where задає точку відліку для зсуву:
SEEK_SET – зсув від початку файлу, SEEK_CUR – зсув від поточної
```

позиції покажчика, **SEEK_END** – зсув від кінця файлу. Наприклад:
fseek(fp, 10L, SEEK_SET);

Функція установлення покажчика запису-читання на початок файлу: **void rewind(FILE *fp);** // прототип

Наприклад: **rewind(fp);**

Функція визначення поточного положення покажчика запису-читання: **long ftell(FILE *fp);** // прототип

Повертається значення в байтах від початку файлу до поточного положення покажчика запису-читання файлу. Наприклад:

long cur=ftell(fp);

Визначення потокових об'єктів у C++. Для роботи з файлами в C++ в файлі **<fstream.h>** визначені наступні класи:

клас **ifstream** – для введення (читання) даних з файлу;

клас **ofstream** – для виведення (запису) даних у файл;

клас **fstream** – для одночасного файлового введення/виведення.

Для роботи з файлами спочатку визначається потоковий об'єкт:

ofstream outFile; // вихідний файловий потік (для запису)

ifstream inFile; // вхідний файловий потік (для читання)

fstream ioFile; // файловий потік для введення/виведення

Після визначення файлового потоку його приєднують до конкретного файлу за допомогою метода **open()**.

outFile.open("test1.txt"); // відкриття потоку для запису

inFile.open("test1.txt"); // відкриття потоку для читання

Два цих етапи (визначення потокового об'єкту та його приєднання до файлу) можна об'єднати:

// визначено об'єкт та приєднано до файлу **"test1.txt"**

ofstream outFile("test1.txt");

ifstream inFile("test1.txt");

Для перевірки правильності відкриття та приєднання файлу використовується перевантажена операція !

if(!outFile) {cerr<<"Error:\n"; exit(1);}

if(!inFile) {puts<<"Error:\n"; return 1;}

Зв'язок з файлом завершується автоматично після виконання програми, або його можна перервати явно по закінченню обміну даними за допомогою метода **close()**.

outFile.close(); // перервати зв'язок з файлом для запису

inFile.close(); // перервати зв'язок з файлом для читання

Методи запису в файл та читання з файлу. В класі **ofstream** для запису даних у файл використовується операція включення **<<**. При цьому виконується перетворення даних в символьний вигляд.

```
int a1=5; double b1=123.456;
char ch1='W', str1[]="abcdefgh";
ofstream<<a1<<b1; // запис цілого та дійсного чисел
ofstream<<ch1<<str1; // запис символу та рядка
```

В класі **ifstream** для читання даних з файлу використовується операція вилучення **>>**. При цьому виконується перетворення даних з символьного вигляду в заданий тип.

```
int a2; double b2; char ch2, str2[20];
ifstream>>a2>>b2; //читання з файлу цілого і дійсного чисел
ifstream>>ch2>>str2; //читання з файлу символу та рядка
//виведення прочитаних змінних на екран
cout<<a2<<" "<<b2<<endl;
cout<<ch2<<" "<<str2<<endl;
```

В класі **ofstream** для неформатованого запису даних у файл використовуються методи **put(sim)** та **write()**.

Метод **put(sim)** записує символ **sim** в файл.

```
char ss='F';
ofstream.put(ss); // запис в файл символу
```

Метод **write(buf,num)** записує **num** символів з символьного масиву **buf** в файл. При цьому не виконується автоматичне перетворення даних. Для запису даних, що не є символьним масивом, потрібне явне перетворення типу.

```
char ch3='Z', str3[]="New file";
ofstream.write(&ch3,1); // запис символу
ofstream.write(str3,strlen(str3)); // запис рядка
```

Запис в файл чисел з перетворенням типу:

```
int a3=76; double b3=876.12;
ofstream.write((char*)(&a3),sizeof(int));
ofstream.write((char*)(&b3),sizeof(double));
```

В класі **ifstream** для неформатованого читання даних з файлу використовуються метод **read()**. При цьому не виконується автоматичне перетворення даних. Для читання даних, що не є символьним масивом, потрібне явне перетворення типу.

```
char ch4, str4[40];
ifstream.read(&ch4,1); //читання з файлу символу
ifstream.read(str4,40); //читання з файлу рядка
```

Читання з файлу чисел з перетворенням типу:

```
int a2; double b2;
ifstream.read((char*)(&a4),sizeof(int));
ifstream.read((char*)(&b4),sizeof(double));
```

В класі `ifstream` для читання з файлу символів, рядків (символьних масивів) використовуються ще методи `get()`, `getline()`.

Метод `get(sim)` зчитує символ з файлу і записує в змінну `sim`.

```
char ch5; get(ch5); cout<<ch5<<endl;
```

Метод `get(buf,num,lim='\n')` зчитує `num-1` символів (або поки не зустрівся символ `'\n'`), але залишає символ-роздільник рядків `'\n'` у потоці, тому необхідно ще раз викликати метод `get()` для пропуску цього символу.

```
char str5[40];
ifstream.get(str5,40); // читання з файлу рядка без '\n'
ifstream.get(); // читання з файлу '\n'
```

Метод `getline()` зчитує рядок з файлу та вилучає символ-роздільник рядків `'\n'` з потоку.

```
char str6[40];
ifstream.getline(str6,40); // читання з файлу рядка з '\n'
```

Метод `gcount()` повертає кількість символів, прочитаних останню функцією неформатованого введення.

```
int istr=inFile.gcount(); // кількість прочитаних символів,
```

Довільний доступ до файлу. В класах `ofstream` та `ifstream` визначені наступні методи довільного доступу до файлу:

Метод `seekg(pos)` встановлює поточну позицію запису (читання) в значення `pos`.

Метод `seekg(off,org)` переміщує поточну позицію запису (читання) на `offs` байтів від трьох позицій, що визначаються параметром `org`: `ios::beg` (від початку файлу), `ios::cur` (від поточної позиції) або `ios::end` (від кінця файлу).

Метод `tellg()` повертає поточну позицію запису (читання) потоку.

```
ofstream.seekg(0); //перейти до початку файлу
ofstream.seekg(40,ios::beg); //довільний доступ від початку файлу
ifstream.seekg(0,ios::end); //довільний доступ до кінця файлу
int N=ifstream.tellg(); //поточна позиція читання в потоці
cout<<"поточна позиція читання потоку="<<N<<endl;
```

ЛАБОРАТОРНА РОБОТА

Робота з файлами

Мета роботи – вивчити основні функції роботи з файлами.

Програма 1. Показано роботу функцій відкриття файлу для запису і читання та функцій запису даних. в файл і читання з файлу даних. Після виконання програми перевіряється наявність та зміст файлу `test1.txt` у своєму каталозі..

```
#include <stdio.h>
#include<iostream>
#include <stdlib.h>
int main()
{FILE *fin, *fout;
  const int N=30;
  char filename[12]="test1.txt"; //назва файлу
  fout=fopen(filename, "w"); //відкриття потоку для запису
  if(fout==NULL) //перевірка відкриття файлу
    {puts("Error"); return 1; }
  char ch1='W'; // визначення та ініціалізація символу
  char str1[N]; // визначення рядка
  int a1=1234; // визначення та ініціалізація змінної типу int
  double b1=98.765; // визначення та ініціалізація змінної типу double
  fputc(ch1, fout); // запис символу в файл
  fputc('\n',fout); // запис '\n' в файл
  printf("Input string <= %d simbols\n",N);
  gets(str1); // введення рядка з клавіатури
  fputs(str1,fout); // запис рядка в файл
  fputs("\n",fout); // запис "\n" в файл
  fprintf(fout,"%d\n",a1); //запис в файл числа типу int
  fprintf(fout,"%0.3lf\n",b1); //запис в файл числа типу double
  fclose(fout); // закриття потоку

  char ch2, str2[N+1]; // резервування пам'яті для символу, рядка
  int a2; double b2; //резервування пам'яті для чисел типу int,double
  fin=fopen(filename, "r"); //відкриття потоку для читання
  if(fin==NULL) //перевірка відкриття файлу
    {puts("Error "); return 2; }
  puts("Simbol in file:");
  ch2=fgetc(fin); // читання символу з файлу
```

```

putchar(ch2); // виведення символу на екран
fgetc(fin); // читання '\n' з файлу
puts("\nString in file:");
fgets(str2,N+1,fin); // читання рядка з файлу не більш N байтів
puts(str2); // виведення рядка на екран
puts("Number in file:");
fscanf(fin,"%d\n",&a2); // читання з файлу числа типу int
fscanf(fin,"%lf\n",&b2); // читання з файлу числа типу double
printf("%d\n",a2); // виведення на екран числа типу int
printf("%.3lf\n",b2); // виведення на екран числа типу double
fclose(fin); // закриття потоку
system("pause");
return 0; }

```

Завдання до програми 1

Розробіть програму, що відкриває файл з іменем, що складається з перших чотирьох букв Вашого прізвища з розширенням **txt**. Запишіть задані згідно варіанту символи у відкритий файл за допомогою функції **fputc()**, рядки за допомогою функції **fputs()** та числа за допомогою функції **fprintf()**. Перевірте правильність запису, для цього прочитайте з файлу записані символи за допомогою функції **fgetc()**, рядки за допомогою функції **fgets()** та числа за допомогою функції **fscanf()** та виведіть їх на екран.

1. Запишіть у файл посимвольно перші шість букв латинського алфавіту. Визначте масив рядків та проініціалізуйте його назвами дисциплін, з яких треба скласти іспити, і запишіть рядки у файл. Визначте та проініціалізуйте масив з 4-х чисел типу **int**, і запишіть масив чисел у файл. Перевірте правильність запису.

2. Визначте масив з 3-х символів, введіть їх з клавіатури і запишіть їх посимвольно у файл. Визначте масив рядків та проініціалізуйте Вашими прізвищем, ім'ям та по батькові, і запишіть рядки у файл. Визначте та проініціалізуйте число типу **float**, запишіть число в файл. Перевірте правильність запису.

3. Визначте символ, введіть його з клавіатури і запишіть його у файл. Визначте масив з 3-х рядків і проініціалізуйте його назвами книг, та запишіть масив рядків у файл. Визначте і проініціалізуйте масив з 4-х чисел типу **double**, запишіть масив чисел у файл. Перевірте правильність запису.

4. Введіть з клавіатури символи до першого введення символу 'q' та запишіть їх у файл. Визначте масив рядків і проініціалізуйте назвами дисциплін, з яких треба скласти залік, запишіть масив рядків у файл. Визначте та проініціалізуйте число типу **long**, запишіть число в файл. Перевірте правильність запису.
5. Визначте символ, який проініціалізуйте першою буквою Вашого ім'я, і запишіть у файл. Задайте кількість рядків. Введіть з клавіатури задану кількість рядків та запишіть їх у файл. Визначте та проініціалізуйте число типу **float** та число типу **double**, запишіть числа в файл. Перевірте правильність запису.

Програма 2. Показана робота з файлами в C++. Для запису даних у файл використовується операція включення <<, а для читання даних з файлу використовуються операція вилучення >>.

```
#include <stdlib.h>
#include <iostream>
#include <fstream> //підключення бібліотеки для роботи з файлами
using namespace std;

int main()
{ofstream ofFile; //визначення потоку для запису ofFile
  ofFile.open("file2.txt"); //відкриття файлу для запису
  if(!ofFile) //перевірка відкриття файлу
    {cerr<<"Error:\n"; exit(1);}
// визначення та ініціалізація змінних
  int a=40;
  double d=123.45;
  char s='m';
  char str[]="winter";
//запис змінних у файл
  ofFile<<a<<endl;
  ofFile <<d<<endl;
  ofFile<<s<<endl;
  ofFile<<str<<endl;
  ofFile.close(); //закриття файлу

  ifstream ifFile; // визначення потоку для читання ifFile
  ifFile.open("file2.txt");// відкриття файлу для читання
  if(!ifFile) //перевірка відкриття файлу
    {cerr<<"Error:\n"; exit(1);}
```

```
// резервування пам'яті для змінних
int a1; double d1;
char s1; char str1[20];
// читання змінних з файлу
ifstream>>a1>>d1>>s1>>str1;
//виведення прочитаних змінних на екран
cout<<a1<< endl<<d1<< endl;
cout<<s1<< endl;
cout<<str1<<endl;
ifstream.close(); // закриття файлу
return 0;
}
```

Завдання до програми 2

Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть дані, задані згідно варіанту, у відкритий файл за допомогою операції вставки <<. Перевірте правильність запису, для цього прочитайте з файлу записані дані за допомогою операції вилучення >> та виведіть їх на екран.

1. Визначте символ, значення якого введіть з клавіатури, визначте і проініціалізуйте масив з чотирьох чисел типу **int**, визначте змінну типу **float**, яку проініціалізуйте будь-яким значенням. Запишіть дані у файл. Перевірте правильність запису.
2. Визначте символ, який проініціалізуйте першою буквою імені, визначте і проініціалізуйте масив з п'яти чисел типу **float**, визначте рядок, який введіть з клавіатури. Запишіть дані у файл. Перевірте правильність запису.
3. Визначте масив з трьох чисел типу **int**, значення якого введіть з клавіатури, визначте змінну типу **double** і рядок, які проініціалізуйте будь-якими значеннями. Запишіть дані у файл. Перевірте правильність запису.
4. Визначте і проініціалізуйте масив з чотирьох чисел типу **float**, визначте символ та змінну типу **int**, значення яких введіть з клавіатури. Запишіть дані у файл. Перевірте правильність запису.
5. Визначте і проініціалізуйте рядок, визначте символ та масив з трьох чисел типу **int**, значення яких введіть з клавіатури. Запишіть дані у файл. Перевірте правильність запису.

Програма 3. Показана робота з файлами в C++. Для запису даних у файл використовується функція **write()**, а для читання даних з файлу використовуються функції **read()**, **get()**

```
#include <stdlib.h>
#include <iostream>
#include <fstream>    // бібліотека для роботи з файлами
#include <string.h>    // бібліотека для роботи з рядками
using namespace std;
int main(){
// визначення потоку для запису out і зв'язування його з файлом
ofstream out("test3_ck.txt");
if(!out) //перевірка відкриття файлу
{ cerr<<"Error:\n"; return 1;}
// визначення та ініціалізація змінних і рядків
double num1=100.45;
int num2=1234;
char str1[]="Winter has come. ";
char str2[]="Session coming soon.";
//запис в файл змінних з перетворенням в рядок
out.write((char*) (&num1),sizeof(double));
out.write((char*) (&num2),sizeof(int));
//запис в файл рядків
out.write(str1, strlen(str1));
out.write(str2, strlen(str2));
out.close(); // закриття файлу
// визначення потоку для запису in і зв'язування його з файлом
ifstream in("test3_ck.txt");
if(!in) //перевірка відкриття файлу
{ cerr<<"Error:\n"; return 1;}
// резервування пам'яті для змінних та рядка
double num1_c;
int num2_c, istr;
char str1_c[60]; char str2_c[60];
// читання з файлу рядків з перетворенням в заданий тип
in.read((char*) (&num1_c),sizeof(double));
in.read((char*) (&num2_c),sizeof(int));
// читання з файлу рядків
in.read(str1_c, strlen(str1));
istr=in.gcount(); // кількість прочитаних символів в рядку
```

```

str1_c[istr]='\0'; // занести нуль-символ у кінець рядка
in.read(str2_c, strlen(str2));
istr=in.gcount(); // кількість прочитаних символів в рядку
str2_c[istr]='\0'; // занести нуль-символ у кінець рядка
//виведення прочитаних змінних на екран
cout<<num1_c<<' '<<num2_c<<endl;
cout<<str1_c << endl;
cout<<str2_c << endl;
in.close(); //закриття файлу
return 0;}

```

Завдання до програми 3

Розробіть програму, що відкриває файл з іменем, складається з перших літер Вашого прізвища, імені, по батькові. Запишіть дані, задані згідно варіанту, у відкритий файл за допомогою функції **write()**. Перевірте правильність запису, для цього прочитайте з файлу записані дані за допомогою функцій **read()** або **getline()** та виведіть їх на екран.

1. Визначте і проініціалізуйте масиви з такими даними про працівника: порядковий номер (ціле число), прізвище (рядок), рік народження (ціле число), стаж роботи (дійсне число). Запишіть дані про чотирьох працівників. Перевірте правильність запису.
2. Визначте і проініціалізуйте масиви з такими даними про студента: порядковий номер (ціле число), прізвище (рядок), рік народження (ціле число), форма навчання бюджет/контракт (символ). Запишіть дані про чотирьох студентів. Перевірте правильність запису.
3. Визначте і проініціалізуйте масиви з такими даними про прогноз погоди: дата (рядок), температура (ціле число), тиск (ціле число), ймовірність опадів (дійсне число). Запишіть дані про прогноз погоди на чотири дні. Перевірте правильність запису.
4. Визначте і проініціалізуйте масиви з такими даними про книги: порядковий номер (ціле число), назва книги (рядок), прізвище автора (рядок), рік видання (ціле число), кількість примірників в бібліотеці. Запишіть дані про п'ять книг. Перевірте правильність запису.
5. Визначте і проініціалізуйте масиви з такими даними про дисципліни в семестрі: порядковий номер (ціле число), назва дисципліни (рядок), прізвище викладача (рядок), форма контролю іспит/залік (символ), наявність курсової роботи 0/1 (ціле число). Запишіть дані про п'ять дисциплін. Перевірте правильність запису.

КОНТРОЛЬНІ ПИТАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

1. Що означає запис **FILE *fp**? Як відкрити потік **fp** для запису даних і як закрити цей потік?
2. Для чого призначені функції **fputc()** та **fgetc()**? Приведіть приклади їх використання.
3. Для чого призначені функції **fputs()** та **fgets()**? Приведіть приклади їх використання.
4. Для чого призначені функції **fprintf()** та **fscanf()**? Приведіть приклади їх використання.
5. Для чого призначені функції **fwrite()** та **fread()**? Приведіть приклади їх використання.
6. Для чого призначені функції **rewind()**, **fseek()**, **ftell()**? Приведіть приклади їх використання.
7. Які визначені класи для роботи з файлами? Для чого вони призначені?
8. Як створити об'єкт класу **ofstream** і приєднати його до файлу? Як перевірити правильність відкриття та приєднання файлу?
9. Як записати дані в класі **ofstream** з використанням операції вклучення **<<** або функції **write()**? Приведіть приклади.
10. Як прочитати дані в класі **ifstream** з використанням операції вилучення **>>** або функцій **read()**? Приведіть приклади
11. Для чого призначені методи **get()**, **getline()**? В чому різниця? Приведіть приклади
12. Для чого призначені функції **tellg()**, **seekg(pos)**, **seekg(off,org)**? Приведіть приклади їх використання

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Подбельский В.В. Язык Си++: Учеб. пособие. – М.: Финансы и статистика, 2007. – 560 с.
2. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учебн. пособие. – 2-е изд., – М.: Финансы и статистика, 2009. – 600с.
3. Павловская Т.А. С/С++. Программирование на языке высокого уровня. – СПб: Питер, 2009. – 464 с
4. Крупник А. Изучаем Си. – СПб.: Питер, – 2001. – 256с.
5. Стивен Прата. Язык программирования С++. Лекции и упражнения. – 6-е изд. / Пер. с англ. – М.: Вильямс, 2012. – 478 с.
6. С/С++. Структурное программирование. Практикум / Т.А. Павловская, Ю.А. Щупак. – СПб.: Питер, 2002. – 240с.
7. Х.М. Дейтел, П.Дж. Дейтел. Как программировать на С++. – 5-е изд. / Пер. с англ. – М.: ООО "Бином-Пресс", 2008.– 1456 с.
8. Шилдт, Герберт. Полный справочник по С++. – 4-е изд. / Пер. с англ. – М.: Вильямс, 2008. – 800 с.
9. Брайан Керниган, Деннис Ритчи. Язык программирования С. – 2-ое изд. / Пер. с англ. – М.: Вильямс, 2009. – 532 с.
10. Васильев А.К. Программирование на С++ в примерах и задачах. – СПб: Питер, 2017. –369 с.
11. Страуструп Б. Программирование. Принципы и практика с использованием С++. – 2 изд. / Пер. с англ. – М.: Вильямс, 2016. – 1329с.