

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
«НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»  
Інститут телекомунікацій, радіоелектроніки та електронної техніки  
Кафедра «Радіоелектронні пристрої та системи»



Звіт з лабораторної роботи № 19  
«Програмування, частина 2»

Підготував:  
ст. гр. IX-11  
Диркавець Максим  
Перевірив:  
Асистент каф РЕПС  
Чайковський  
І.Б.

Львів 2024

**Тема:** Дослідження способів організації потокового уведення/виведення в мові програмування C.

**Мета роботи:** Дослідження способів створення, оновлення та оброблення файлів потокового уведення/виведення даних у мові C.

### Хід роботи

1. Дослідити та дати пояснення прикладів, викладених нижче

Приклад а

```
#include <stdio.h>

int main() {
    FILE *in;
    int ch;
    if ((in = fopen("proba", "r")) != NULL) {
        while ((ch = getc(in)) != EOF) {
            putc(ch, stdout);
        }
        fclose(in);
    } else {
        printf("Файл proba не відкривається\n");
    }
    return 0;
}
```

Файл proba не відкривається

### Пояснення 1

Відкривається файл з назвою "proba" для читання.

Якщо файл успішно відкрився, тобто **fopen("proba", "r")** повернуло не NULL, програма починає зчитувати символи з файлу за допомогою функції **getc(in)**.

Кожен зчитаний символ виводиться на екран за допомогою **putc(ch, stdout)**.

Ця операція повторюється до тих пір, поки не буде досягнута кінцева позиція

файлу, що вказує на кінець файлу (EOF).

Після закінчення зчитування і виведення файл закривається за допомогою **fclose(in)**.

Результатом виконання програми буде виведення вмісту файлу "proba" на екран

Приклад б

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

```
int main() {
```

```
    FILE *ff;
```

```
    int base;
```

```
    ff = fopen("sam", "r");
```

```
    if (ff == NULL) {
```

```
        printf("Pomyłka: ne mozhna vidkryty fail\n");
```

```
        return 1;
```

```
    }
```

```
    fscanf(ff, "%d", &base);
```

```
    fclose(ff);
```

```
    ff = fopen("data", "a");
```

```
    if (ff == NULL) {
```

```
        printf("Pomyłka: ne mozhna vidkryty fail\n");
```

```
        return 1;
```

```
    }
```

```
    fprintf(ff, "sam is %d.\n", base);
```

```
    fclose(ff);
```

```
    return 0;
```

```
}
```

```
Pomyłka: ne mozhna vidkryty fail
```

## Пояснення 1

Відкривається файл "sam" для читання.

Якщо файл вдалося відкрити, програма зчитує число з файлу "sam".

Після завершення читання файл "sam" закривається.

Відкривається файл "data" для дописування.

Якщо файл вдалося відкрити, програма записує рядок у файл "data", який містить число, зчитане з файлу "sam".

Файл "data" закривається.

Якщо відкриття файлу не вдасться, програма виведе повідомлення про помилку та завершиться з кодом 1.

## Приклад в

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

```
#define LINE 80
```

```
int main() {
```

```
    FILE *ff;
```

```
    char string[LINE];
```

```
    ff = fopen("opus", "r");
```

```
    if (ff == NULL) {
```

```
        printf("Pomyłka: nie mozhna vidkryty fail\n");
```

```
        return 1;
```

```
    }
```

```
    while (fgets(string, LINE, ff) != NULL) {
```

```
        puts(string);
```

```
    }
```

```
    fclose(ff);
```

```
    return 0;
```

```
}
```

```
Pomyłka: ne mozhna vidkryty fail
```

#### Пояснення 1

Відкривається файл "opus" для читання.

Якщо відкриття файлу успішне, програма у циклі зчитує рядки з файлу, використовуючи функцію **fgets**.

Зчитаний рядок виводиться на екран за допомогою функції **puts**.

Цикл продовжується до тих пір, поки не буде досягнуто кінця файлу або не виникне помилка під час читання.

Після завершення зчитування файл закривається.

#### Приклад г

```
#include <stdlib.h>
```

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

```
int main() {
```

```
    int f1, f2, f3, f4, f5;
```

```
    FILE *fp;
```

```
    fp = fopen("C:\\temp\\sample.txt", "r");
```

```
    if (fp == NULL) {
```

```
        printf("Pomyłka: ne mozhna vidkryty fail\n");
```

```
        return 1;
```

```
    }
```

```
    fscanf(fp, "%d\n%d\n%d\n%d\n%d\n", &f1, &f2, &f3, &f4, &f5);
```

```
    printf("The values are %d, %d, %d, %d, %d.\n", f1, f2, f3, f4, f5);
```

```
    fclose(fp);
```

```
    return 0;
```

```
}
```

**Pomylka: ne mozhna vidkryty fail**

#### Пояснення 1

Програма відкриває файл "C:\temp\sample.txt" в режимі читання.

Якщо відкриття файлу успішне, програма зчитує з файлу п'ять цілих чисел за допомогою функції **fscanf**.

Зчитані значення виводяться на екран за допомогою функції **printf**.

Після завершення роботи з файлом він закривається за допомогою функції **fclose**.

2.Розглянути функції форматного обміну з файлами **fprintf()**, **fscanf()** пояснити їх відмінності від функцій **printf()**, **scanf()**.

Функції **fprintf()** та **fscanf()** - це як **printf()** та **scanf()**, але для роботи з файлами. Коли ми хочемо записати щось у файл за вказаним форматом, ми використовуємо **fprintf()**. Наприклад, якщо ми хочемо записати число у файл, ми використовуємо **fprintf()**. А якщо ми хочемо прочитати дані з файлу за вказаним форматом, ми використовуємо **fscanf()**. Вони дозволяють нам читати та записувати дані у файл у зручний для нас спосіб, використовуючи ті самі специфікатори формату, які ми використовуємо для **printf()** та **scanf()**.

3.Виконати програму, що створює файл proba.txt і записує в нього символні зображення чисел від 0 до 5 і їх кубів. Наступною програмою прочитати дані із файлу proba.txt. У звіті дати детальне пояснення роботи програм.

#### Програма 1

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

```
int main(void)
```

```
{
```

```
    FILE *pf;
```

```
    int k;
```

```
    if ((pf = fopen("proba.txt", "w")) == NULL) {
```

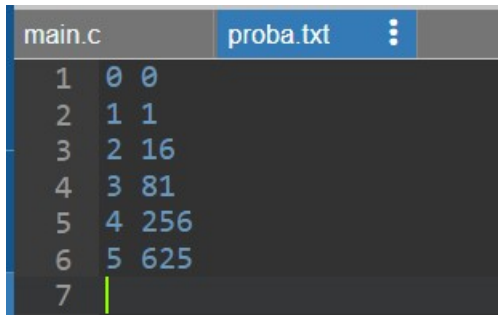
```
        perror("proba.txt");
```

```
    }
```

```

for (k = 0; k <= 5; k++) {
    fprintf(pf, "%d %d\n", k, k * k * k * k);
}
fclose(pf);
return 0;
}

```



```

main.c  proba.txt  ⋮
1  0  0
2  1  1
3  2  16
4  3  81
5  4  256
6  5  625
7

```

Пояснення:

Включення бібліотеки **<stdio.h>**, що містить визначення функцій для роботи з файлами та ввідно-вивідом.

Об'явлення функції **main()**, яка є головною функцією програми.

Об'явлення змінних:

**pf** - вказівник на файл.

**k** - змінна для лічильника у циклі.

Спроба відкрити файл "proba.txt" для запису за допомогою функції **fopen()**.

Якщо відкриття не вдалося (у випадку, коли файл відсутній або немає прав на запис), виводиться помилка за допомогою **perror()** та програма завершується з кодом помилки 1.

Цикл **for** від 0 до 5, в якому виконується запис чисел та їх кубів у файл за допомогою функції **fprintf()**.

Закриття файлу за допомогою функції **fclose()**.

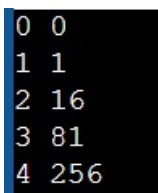
Повернення 0, що показує, що програма завершилася успішно.

## Програма 2

```
#include <stdio.h>

int main(void) {
    FILE *pf;
    int n, nn, l;
    if ((pf = fopen("proba.txt", "r")) == NULL) {
        perror("proba.txt");
        return 1;
    }
    for (l = 0; l < 5; l++) {
        fscanf(pf, "%d %d\n", &n, &nn);
        printf("%d %d\n", n, nn);
    }
    fclose(pf);

    return 0;
}
```



```
0 0
1 1
2 16
3 81
4 256
```

### Пояснення:

Функція **main()** - головна функція програми, в якій виконується основний алгоритм програми.

Оголошуються змінні: **pf** - вказівник на файл, **n**, **nn** - числа, які будуть зчитані з файлу, **l** - лічильник циклу.

Відкривається файл "proba.txt" для читання у режимі "r" за допомогою функції **fopen()**. Перевіряється, чи відбулося відкриття файлу.



В циклі **for** зчитуються дані з файлу у пари чисел за допомогою функції **fscanf()**. Кожна пара чисел записується у змінні **n** та **nn**.

Зчитані числа виводяться на екран за допомогою функції **printf()**.

Файл закривається за допомогою функції **fclose()**.

Функція **main()** повертає 0, що означає успішне завершення програми.

**Висновок:** Під час даної роботи ми дослідили способи організації потокового уведення та виведення даних у мові програмування C. Ми ознайомилися з функціями стандартної бібліотеки мови C, такими як `'fopen()'`, `'fclose()'`, `'fprintf()'`, `'fscanf()'`, `'fgets()'`, `'fputs()'`, та з їхнім використанням для створення, оновлення та оброблення файлів. Використання цих функцій дозволяє нам ефективно працювати з файлами у мові C, забезпечуючи потрібний рівень контролю та обробки даних.