

**Міністерство освіти і науки України**  
**Національний університет “Львівська політехніка”**



**Лабораторна робота №19**  
**з дисципліни**  
**«Програмування частина 2»**

**Виконав:**  
Студент групи АП-11  
Братейко Вадим

**Прийняв:**  
Чайковський І.Б.

Львів 2024

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

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

#### Теоретичні відомості

Зберігання даних у змінних і масивах є тимчасовим; всі ці дані втрачаються при завершенні роботи програми. Для постійного зберігання великих об'ємів даних використовуються файли. Комп'ютери зберігають дані на пристроях вторинної пам'яті, головним чином дискових пристроях.

Комп'ютер обробляє елементи даних в двійковому вигляді, тобто у вигляді комбінацій нулів і одиниць.

Для програмістів обтяжливо працювати з даними низького рівня, якими є біти. Замість цього програмісти вважають за краще працювати з даними у вигляді десяткових цифр, букв, і спеціальних знаків, які називаються символами.

Стандартна бібліотека підтримує численні функції читання даних з файлів і запису даних у файли. Функція `fgetc`, подібно `getchar`, прочитує з файлу один символ. Функція `fgetc` отримує як аргумент вказівник на `FILE` для файлу, з якого прочитуватиметься символ. Виклик `fgetc(stdin)` читає один символ з `stdin` – стандартного уведення. Такий виклик еквівалентний виклику `getchar()`. Функція `fputc`, подібно `putchar`, записує один символ у файл. Як аргумент функція отримує символ, який повинен бути записаний. Виклик функції – `fputc('a', stdout)` записує символ 'a' в `stdout` – стандартний вивід. Такий виклик цієї функції еквівалентний `putchar('a')`.

Отже мова С розглядає файл як структуру з ім'ям шаблону (тегом) `FILE`, що вказує на ім'я файлу, його статус та текучу позицію. Режими функції `fopen`

– знаходяться в `stdio.h`:

“r” – відкрити для читання (файл має існувати);

“w” – створити для запису (якщо файл існує, вміст губиться);

“a” – відкрити для додавання в існуючий файл (файл створюється, якщо не існує);

“rb” – відкрити двійковий файл для читання;

“wb” – відкрити двійковий файл для запису;

“r+” – відкрити файл для читання і запису (файл має існувати);

“w+” – створити файл для читання і запису (якщо файл існує, вміст губиться).

#### Приклад 1

//a

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

```
int main() {
```

```
FILE *in; // Опис вказівника на файл
```

```
int ch;
```

```
if ((in = fopen("proba", "r")) != NULL) { // Відкриття файлу для читання, перевірка чи існує
```

```

while ((ch = getc(in)) != EOF) { // Отримання символу із файла
putc(ch, stdout); // Виведення символу в стандартний вивід
}
fclose(in); // Закриття файла
} else {
printf("Файл proba не відкривається\n");
return 0;}

```

**//б**

```

#include <stdio.h>
int main() {
    FILE *ff;
    int base;
    ff = fopen("sam", "r"); // Відкриття файла для читання
    fscanf(ff, "%d", &base); // Читання значення з файла
    fclose(ff); // Закриття файла
    ff = fopen("data", "a"); // Відкриття файла для дописування
    fprintf(ff, "sam is %d.\n", base); // Запис значення у файл
    fclose(ff); // Закриття файла
    return 0; }

```

**//в**

```

#include <stdio.h>
#define LINE 80
int main() {
    FILE *ff;
    char string[LINE]; // Масив для зберігання рядка
    ff = fopen("opus", "r"); // Відкриття файла для читання
    while (fgets(string, LINE, ff) != NULL) { // Зчитування рядків з файла
        puts(string); // Виведення рядка на екран
    }
    fclose(ff); // Закриття файла
    return 0;}

```

**//г**

```

#include <stdlib.h>
#include <stdio.h>
int main() {
    int f1, f2, f3, f4, f5;
    FILE *fp;
    fp = fopen("C:\\temp\\sample.txt", "r"); // Відкриття файла для читання
    if (fp == NULL) { // Перевірка чи файл відкрито успішно
        printf("Помилка відкриття файла\n");
        return 1;}
    // Читання з файла
    fscanf(fp, "%d\n%d\n%d\n%d\n%d\n", &f1, &f2, &f3, &f4, &f5);
    printf("Значення: %d, %d, %d, %d, %d\n", f1, f2, f3, f4, f5);
}

```

```
fclose(fp); // Закриття файлу
return 0;}
```

## Приклад 2

Функції форматного обміну з файлами `fprintf()` і `fscanf()` у мові C дозволяють записувати та зчитувати дані з файлів у відповідності з вказаними форматами, аналогічно функціям `printf()` і `scanf()`, які працюють зі стандартним введенням та виведенням.

**Отже, ось їх відмінності:**

**`fprintf()` та `fscanf()`:**

`fprintf()`: Ця функція використовується для запису даних у файл з використанням заданих форматів. Вона приймає файловий вказівник та список аргументів, аналогічно `printf()`.

`fscanf()`: Ця функція використовується для зчитування даних з файлу за заданими форматами. Вона приймає файловий вказівник та список аргументів, аналогічно `scanf()`.

**Відмінності від `printf()` та `scanf()`:**

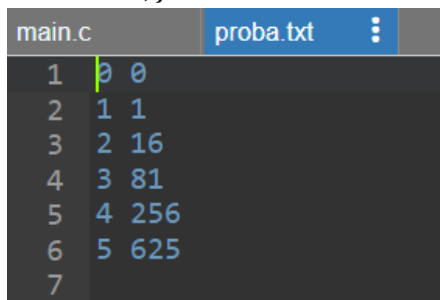
`printf()` та `scanf()`: Ці функції працюють зі стандартним введенням та виведенням (консоль).

`fprintf()` та `fscanf()`: Ці функції працюють з файлами, що дозволяє записувати та зчитувати дані з файлів.

## Приклад 3

```
// Запис файлу
#include <stdio.h>
int main() {
    FILE *pf; // Вказівник на файл
    int k;
    // Відкриття файлу для запису
    if ((pf = fopen("proba.txt", "w")) == NULL) { // Перевірка чи відкриття файлу
        perror("proba.txt"); // Виведення повідомлення про помилку
        return 1; // Повернення коду помилки
    }
    // Запис даних у файл
    for (k = 0; k <= 5; k++) {
        fprintf(pf, "%d %d\n", k, k*k*k*k); // Запис значень у файл
    }
    fclose(pf); // Закриття файлу
}
```

```
return 0;}
```



1	0	0
2	1	1
3	2	16
4	3	81
5	4	256
6	5	625
7		

```
// Читання даних із файлу proba.txt
#include <stdio.h>
int main() {
    FILE *pf; // Вказівник на файл
    int n, nn, l;
    // Відкриття файлу для читання
    if ((pf = fopen("proba.txt", "r")) == NULL) { // Перевірка чи відкриття файлу
        perror("proba.txt"); // Виведення повідомлення про помилку
        return 1; } // Повернення коду помилки
    }
    // Читання даних з файлу
    for (l = 0; l <= 5; l++) {
        fscanf(pf, "%d %d\n", &n, &nn); } // Зчитування значень з файлу
    fclose(pf); // Закриття файлу
    return 0;}
```

**Приклад 4** Задати 8 змінних цілого типу, записати їх в файл на диску, прочитати їх з файлу в змінні і вивести на екран.

```
#include <stdio.h>
void main(){
    int num[8]={ 12,8,4,5,3,8,4,32 };
    FILE *pf;
    int i;
    if ((pf= fopen ("num.txt","w"))==NULL)
    {
        perror("proba.txt");
        return 1; }
    for (i=0; i<8; i++)
        fprintf(pf, "%d\n",num[i]);
    fclose(pf);
```

```
return 0;}
```

main.c	num.txt
1	12
2	8
3	4
4	5
5	3
6	8
7	4
8	32
9	

```
#include <stdio.h>
void main (void) {
FILE *pf;
int a,b;
int n[8];
if ((pf = fopen ("num.txt","r"))== NULL) {
perror("proba.txt");
return 1; }
for (b=0; b<8; b++){
fscanf(pf, "%d\n", &n[b]); }
fclose(pf);
for (a=0; a<8; a++){
printf("%d\n", n[a]); }
return 0;}
```

```
12
8
4
5
3
8
4
32
```

### Відповіді на контрольні запитання

1) Суть поняття файлу в мові програмування C:

У мові програмування C файл - це послідовність даних, яка зберігається на зовнішньому носії, такому як жорсткий диск, флеш-накопичувач або інше зберігання. Файли використовуються для зберігання та обміну даними між програмами та користувачами.

2) Основні режими відкриття файлу в мові C:

В мові C можна відкрити файл у таких режимах:

"r": Відкриття для читання. Файл має існувати.

"w": Відкриття для запису. Якщо файл існує, його вміст буде видалено; якщо файл не існує, він буде створено.

"a": Відкриття для додавання. Дані додаються до кінця файлу. Файл, якщо він існує, не буде видалено.

"r+": Відкриття для читання та запису. Файл має існувати.

"w+": Відкриття для читання та запису. Якщо файл існує, його вміст буде видалено; якщо файл не існує, він буде створено.

3) Основні функції при роботі з файлами в мові C:

Основні функції при роботі з файлами в мові C включають такі:

fopen(): Відкриття файлу.

fclose(): Закриття файлу.

fread(): Читання з файлу.

fwrite(): Запис до файлу.

fseek(): Пошук позиції у файлі.

ftell(): Отримання поточної позиції у файлі.

feof(): Перевірка кінця файлу.

4) Способи позиціювання в файлі в мові C:

У мові C для позиціювання в файлі можна використовувати функцію fseek(), яка дозволяє встановлювати позицію вказівника файлу у відповідності з вказаними параметрами. Параметри fseek() включають:

Початок файлу (SEEK\_SET).

Поточна позиція (SEEK\_CUR).

Кінець файлу (SEEK\_END).

**Висновок:** на цій лабораторній роботі я ознайомився і дослідив способи створення, оновлення та оброблення файлів потокового введення/виведення даних у мові C.