Міністерство освіти і науки України

Національний університет “Львівська Політехніка”

Лабораторна робота №19

Виконав:
Студент групи АП-11
Білий Анатолій Іванович

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.

Подібно до того, як символи складаються з бітів, поля складаються з символів. Поле є групою символів, які передають значення.

Оброблювані комп'ютерами елементи утворюють ієрархію даних, в якій елементи даних стають більші за розміром і складніші за структурою в міру просування від бітів до символів (байтів), полів і так далі.

Існує багато способів організації записів у файлі. Найбільш популярний з них називається послідовним файлом, в якому записи, як правило, зберігаються в порядку, що визначається за певним правилом. У файлі нарахування заробітної плати, наприклад, записи зберігалися б впорядкованими за табельним номером.

Мова С розглядає будь-який файл як послідовний потік байтів. Кожен файл закінчується маркером кінця файлу, або особливим байтом, визначеним у програмі, що працює з файлом. Коли файл відкривається, йому ставиться у

відповідність потік. На початку виконання програми автоматично відкриваються три файли і пов’язані з ними потоки - стандартний ввід, стандартний вивід і стандартна помилка. Потоки забезпечують канали передачі даних між файлами і програмами. Наприклад, стандартний потік уведення дозволяє програмі зчитувати дані з клавіатури, а стандартний потік виведення дозволяє виводити дані на екран. Відкритий файл повертає вказівник на структуру FILE (визначену в stdio.h), яка містить інформацію, що використовується при роботі з файлом. Ця структура включає дескриптор файлу, тобто індекс у масиві операційної системи, званому таблицею відкритих файлів. Кожен елемент масиву містить блок управління файлом, який використовується операційною системою для доступу до конкретного файлу. Для звертання до стандартного уведення/виведення і стандартному потоку помилок слід скористатися вказівниками файлів stdin, stdout і stderr відповідно.

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').**

Перш ніж читати або записувати інформацію в файл його необхідно відкрити за допомогою стандартної бібліотечної функції fopen(). Функція fopen() виконує 2 дії: 1 – відкриває потік і зв’язує файл на диску з цим потоком. 2 – повертає вказівник, що пов’язаний із цим файлом.

Прототип функції:

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

Вказівник на файл оголошується наступним чином:

```
FILE *fout;
```

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).

Якщо функція fopen() не може відкрити файл, вона повертає значення "NULL", що, як ми знаємо, визначене в stdio.h як 0.

Причини не відкриття файлу: – спроба відкриття неіснуючого файлу; запис в цей час на диск; використання забороненого імені.

Для закриття файлу використовується функція fclose(). Функція fclose() повертає 0, якщо файл закрився успішно, і (-1) в противному випадку.

Для довільного доступу в певну позицію файлу використовується функція **fseek( )**.

Функція fseek( ) дозволяє обробляти файл подібно до масиву і безпосередньо досягати будь-якого визначеного байту в файлі, відкритому функцією fopen( ).

Функція переміщує внутрішній вказівник файлу на нове місце в файлі, яке очислюється за зміщенням offset та вказівкою напрямку відліку whence.

Наступна операція уведення/виведення зі вказаним файлом буде виконана починаючи з тієї позиції на яку виконано переміщення.

Прототип функції **fseek** визначається як:

int fseek(FILE *stream, long int offset, int whence);

де **offset** – число байтів від положення **whence**, що задається вказівником **FILE** на файл, в якому ведеться пошук – **stream.** Аргумент **whence** може мати одне з трьох значень – **SEEK_SET,** або значення 0 – початок файлу, **SEEK_CUR** або значення 1– текуча позиція вказівника файлу, **SEEK_END** або значення

2 – кінець файлу. Ці три символічні константи визначені в заголовному файлі **stdio.h.**

ЗАВДАННЯ

1. Дослідити та дати пояснення прикладів, викладених нижче.

а)

```
#include “stdio.h”
int main()
{
FILE *in; //Опис вказівника на файл
int ch;
```

if(in=fopen(“proba”,”r”))!=NULL) // Відкривається файл для читання, перевіряється чи він існує. Вказівник in тепер посилається на структуру FILE, що пов’язана із proba.

```
{
while((ch=getc(in)!=EOF)// Отримується символ із in
putc(ch,stdout);// Виведення символа в стандартний потік на екран.
fclose(in);
}
else
printf(“Файл proba не відкривається \n”);
}
```

б).

```
#include <stdio.h>
int main( )
{
FILE *ff;
int base;
```

ff = fopen(" sam" , " r"); // відкривається файл із іменем sam, який ідентифікується зі вказівником на ff .

```
fscanf( ff, " %d" , &base);   // ff вказує на файл із іменем sam
fclose(ff);
ff = fopen("data", "a");   // доповнення
fprintf( ff, "sam is %d.\n", base);   /* ff вказує на data */
fclose(ff;
}
```

в).

```
#include <stdio.h>
#define LINE 80
int main( )
{
FILE *ff;
char *string[LINE];
ff = fopen(" opus", "r");
while ( fgets(string, LINE, ff) != NULL)
    puts(string);
```

г).

Приклад читання форматованих даних з файлу "C:\\temp\\sample.txt":

```
/* Читання форматованих даних за допомогою функції fscanf(). */
#include <stdlib.h>
#include <stdio.h>
int main(){
  int f1, f2, f3, f4, f5;
  FILE *fp;
fp = fopen("C:\\temp\\sample.txt", "r"); /*Відкриття файлу в режимі читання*/
 /*Читання з файлу */
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);  /* Закриття файлу fp*/}
```

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

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

```
#include <stdio.h>
void main (void)
{
FILE *pf;
int k;
if ((pf = fopen (“proba.txt”,”w”)) ==NULL)
{
        perror(“proba.txt”);
        return 1;
}
for (k=0; k<=5; k++)
fprintf(pf, “%d %d\n”, k, k*k*k*k);
fclose(pf);
return 0;
}
```

Читання даних із файлу proba.txt

```
#include <stdio.h>
void 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);
```

```
fclose(pf);
return 0;
}
```

4. Виконати завдання згідно варіанта

№ вар.	Завдання
1.	Задати масив цілих чисел розмірністю 10, записати його в файл на диску, прочитати його з файлу в інший масив, новий масив вивести на екран.
2.	Задати 8 змінних цілого типу, записати їх в файл на диску, прочитати їх з файлу в масив розмірністю 8, масив вивести на екран.
3.	Задати масив цілих чисел розмірністю 10, записати його в файл на диску, прочитати його з файлу в інший масив, новий масив вивести на екран.
4.	Числа від 1 до 12 записати в файл на диску, прочитати їх з файлу в змінні і вивести їх на екран.
5.	Задати 8 змінних цілого типу, записати їх в файл на диску, прочитати їх з файлу в змінні і вивести на екран.
6.	Задати масив цілих чисел розмірністю 12, записати його в файл на диску, прочитати його з файлу в 12 змінних цілого типу і вивести їх на екран.
7.	Задати 11 змінних цілого типу, записати їх в файл на диску, прочитати їх з файлу в масив розмірністю 11, масив вивести на екран.
8.	Задати 8 змінних цілого типу, записати їх в файл на диску, прочитати їх з файлу в масив розмірністю 8, масив вивести на екран.
9.	Задати масив цілих чисел розмірністю 13, записати його в файл на диску, прочитати його з файлу в 13 змінних цілого типу і вивести їх на екран.
10.	Задати 8 змінних цілого типу, записати їх в файл на диску, прочитати їх з файлу в змінні і вивести на екран.
11.	Числа від 1 до 12 записати в файл на диску, прочитати їх з файлу в змінні і вивести їх на екран.
12.	Задати масив цілих чисел розмірністю 9, записати його в файл на диску, прочитати його з файлу в інший масив, новий масив вивести на екран.
13.	Задати масив цілих чисел розмірністю 12, записати його в файл на диску, прочитати його з файлу в інший масив, новий масив вивести на екран..

№ вар.	Завдання
14.	Задати 10 змінних цілого типу, записати їх у файл на диску, прочитати їх з файлу в масив розмірністю 10, масив вивести на екран.

А)

```
#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;}
```

Файл proba не відкривається

Б)

```
#include <stdio.h>
int main() {
   FILE *ff;
   int base;
   ff = fopen("sam", "r"); // Відкриття файлу для читання
   fscanf(ff, "%d", &base); // Читання значення з файла
   fclose(ff); // Закриття файла
   ff = fopen("data", "a"); // Відкриття файлу 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 файлу 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 файлу для читання
            if (fp == NULL) { // Перевірка чи файл відкрито успішно
                printf("Помилка відкриття файлу\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 файла
            return 0;}
```

Помилка відкриття файлу

Приклад

```
// 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 пройшло успішно
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;}
// 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 пройшло успішно
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 файла
fclose(pf); // Закриття файлу
return 0;}
```

```
 запис в пейл
 0 0
 1 1
 2 16
 3 81
 4 256
 5 625
 читання
 n = 0, nn = 0
 n = 1, nn = 1
 n = 2, nn = 16
 n = 3, nn = 81
 n = 4, nn = 256
  n = 5, nn = 625
```

Завдання варіант№1

```
#include <stdio.h>
#define SIZE 10
void write_to_file(int arr[]) {
FILE *fp;
fp = fopen("numbers.txt", "w");
if (fp == NULL) {
```

```
printf("Ne vdalosya vidkryty fail dlya zapisu.\n"); return;
}
for (int i = 0; i < SIZE; i++) {
fprintf(fp, "%d ", arr[i]);
}
fclose(fp);
}
void read_from_file(int arr[]) {
FILE *fp;
fp = fopen("numbers.txt", "r");
if (fp == NULL) {
printf("Ne vdalosya vidkryty fail dlya chitannya.\n"); return;
}
for (int i=0; i < SIZE; i++) {
fscanf(fp, "%d", &arr[i]);
}
fclose(fp);
}
int main() {
int array1[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int array2[SIZE];
write_to_file(array1);
printf("Masiv uspishno zapisany u fail 'numbers.txt'.\n");
read_from_file(array2);
printf("Masiv, prochytany z failu:\n");
for (int i = 0; i < SIZE; i++) {
}
printf("%d", array2[i]);
printf("\n");
```

```
    return 0;
}
```

```
    Masiv uspishno zapisany u fail 'numbers.txt'.

    Masiv, prochytany z failu:

    1 2 3 4 5 6 7 8 9 10

Process exited after 0.03823 seconds with return value 0
```

Контрольні запитання.

1. Суть поняття файлу в мові програмування С.

В мові програмування С файл є інструментом для зберігання та організації даних на зовнішньому носії, такому як жорсткий диск. Файли дозволяють програмам зберігати інформацію між виконаннями, обмінюватися даними з іншими програмами та працювати з великими обсягами даних. Кожен файл має унікальне ім'я, за допомогою якого програма може звертатися до нього.
У мові С для роботи з файлами використовуються функції та типи даних з бібліотеки stdio.h. Ці функції дозволяють відкривати, читати, записувати та закривати файли. Коли файл відкрито, програма може звертатися до його вмісту, читати дані з нього або записувати дані до нього.

2. Основні режими відкриття файлу в мові С.

мові програмування С для відкриття файлу використовується функція fopen, яка приймає два параметри: шлях до файлу та режим відкриття. Основні режими відкриття файлу в мові С:

"r" (читання): Відкриває файл для читання. Покажчик розміщується на початку файлу. Якщо файл не існує, функція повертає NULL.

"w" (запис): Відкриває файл для запису. Якщо файл вже існує, він буде перезаписаний. Якщо файл не існує, він буде створений.

"a" (дописування): Відкриває файл для запису в кінець файлу (дописування). Якщо файл не існує, він буде створений. Покажчик розміщується в кінці файлу.

"r+" (читання і запис): Відкриває файл для читання і запису. Покажчик розміщується на початку файлу. Файл повинен існувати.

"w+" (читання і запис): Відкриває файл для читання і запису. Якщо файл вже існує, він буде перезаписаний. Якщо файл не існує, він буде створений.

"a+" (читання і дописування): Відкриває файл для читання і

дописування. Покажчик розміщується в кінці файлу. Якщо файл не існує, він буде створений.

3. Основні функції при роботі з файлами в мові С.

Основні функції для роботи з файлами в мові програмування С:

fopen - відкриває файл.

fclose - закриває файл.

fprintf і fscanf - запис і зчитування з файлу.

fputc і fgetc - запис і зчитування символів.

fputs і fgets - запис і зчитування рядків.

rewind і fseek - переміщення покажчика у файлі.

ftell - поточна позиція покажчика

4. Способи позиціювання в файлі в мові С.

fseek: Функція fseek дозволяє переміщати покажчик файлу на певну позицію. Вона приймає три аргументи: покажчик на файл, зсув відносно вказівника, з яким відбувається рухання (SEEK_SET, SEEK_CUR, SEEK_END). Наприклад, fseek(file_ptr, 0, SEEK_SET) переміщує покажчик на початок файлу, а fseek(file_ptr, -10, SEEK_CUR) переміщує покажчик на 10 байтів назад від поточної позиції.

rewind: Функція rewind встановлює покажчик файлу на початок файлу. Вона еквівалентна fseek(file_ptr, 0, SEEK_SET).

ftell: Функція ftell повертає поточну позицію покажчика у файлі в байтах.

Висновок: під час виконання цієї лабораторної роботи я дослідив способи створення, оновлення та оброблення файлів потокового уведення/виведення даних у мові С.