

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут телекомунікації, радіоелектроніки та електронної техніки

кафедра «Радіоелектронні пристрої та системи»
З дисципліни «Програмування частина 2»
Лабораторна робота №19
«Оператори циклу»

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

Підготував
ст.групи АП-11
Василюк Ростислав

Прийняв:
Чайковський І.Б.

Львів 2024р

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.

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 впорядкованими за табельним номером. 6

Мова C 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.

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 введення. 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')`.

Завдання

1.

A) `#include <stdio.h>`

```
int main()
{
    FILE *in;
    int ch;

    if ((in = fopen("proba", "r")) != NULL)
    {
        while ((ch = fgetc(in)) != EOF)
            putchar(ch);
    }
}
```

```

        fclose(in);
    }
    else
    {
        printf("Файл proba не відкривається\n");
    }

    return 0;
}
Б) #include <stdio.h>

int main()
{
    FILE *in;
    if ((in = fopen("proba", "r")) != NULL)
    {
        FILE *ff;
        int base;

        if ((ff = fopen("sam", "r")) != NULL)
        {
            fscanf(ff, "%d", &base);
            fclose(ff);

            if ((ff = fopen("data", "a")) != NULL)
            {
                fprintf(ff, "sam is %d.\n", base);
                fclose(ff);
            }
            else
            {
                printf("Не вдалося відкрити файл 'data' для додавання\n");
            }
        }
        else
        {
            printf("Не вдалося відкрити файл 'sam'\n");
        }

        fclose(in);
    }
    else
    {
        printf("Файл 'proba' не відкривається\n");
    }
}

```

```
    return 0;
}
```

B) #include <stdio.h>

#define LINE 80

```
int main()
{
    FILE *ff;
    char string[LINE];

    if ((ff = fopen("opus", "r")) != NULL)
    {
        while (fgets(string, LINE, ff) != NULL)
        {
            puts(string);
        }
        fclose(ff);
    }
    else
    {
        printf("Не вдалося відкрити файл 'opus'\n");
    }

    return 0;
}
```

Г) #include <stdlib.h>

#include <stdio.h>

int main() {

int f1, f2, f3, f4, f5;

ФАЙЛ *fp;

fp = fopen("C:\\temp\\sample.txt", "r"); /*Відкриття файлу в режимі І
читання*/

/*Читання з файлу */

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 fp*/}
```

2.

а) Функція `fprintf()` використовується для запису форматованого тексту у файл.

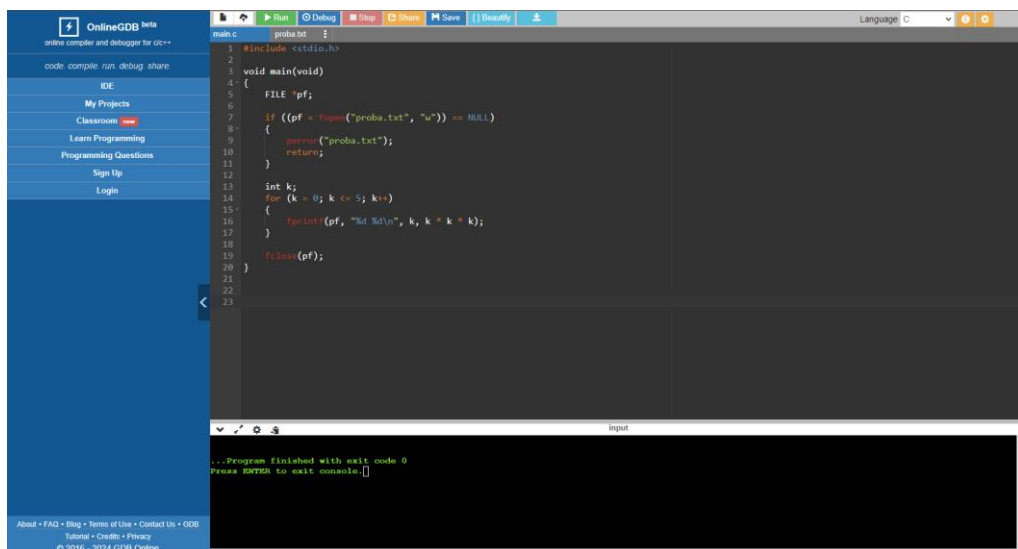
```
int fprintf(FILE *stream, const char *format, ...);
```

б) Функція `fscanf()` використовується для читання форматованого тексту з файлу.

```
int fscanf(FILE *stream, const char *format, ...);
```

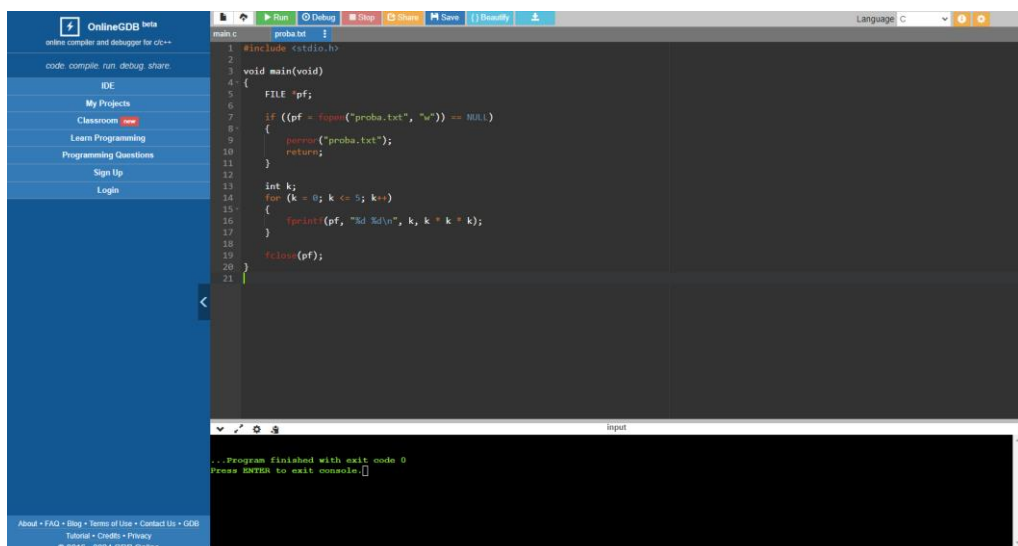
Функції `fprintf()` та `fscanf()` призначені для форматованого вводу-виводу з файлами. Вони схожі на `printf()` та `scanf()`. Основна відмінність між ними полягає у тому, що `fprintf()` та `fscanf()` працюють з файлами, тоді як `printf()` та `scanf()` працюють з стандартними потоками вводу-виводу

3.



```
1 #include <stdio.h>
2
3 void main(void)
4 {
5     FILE *pf;
6
7     if ((pf = fopen("proba.txt", "w")) == NULL)
8     {
9         perror("proba.txt");
10        return;
11    }
12
13    int k;
14    for (k = 0; k <= 5; k++)
15    {
16        fprintf(pf, "%d %d\n", k, k * k * k);
17    }
18    fclose(pf);
19
20
21
22
23
```

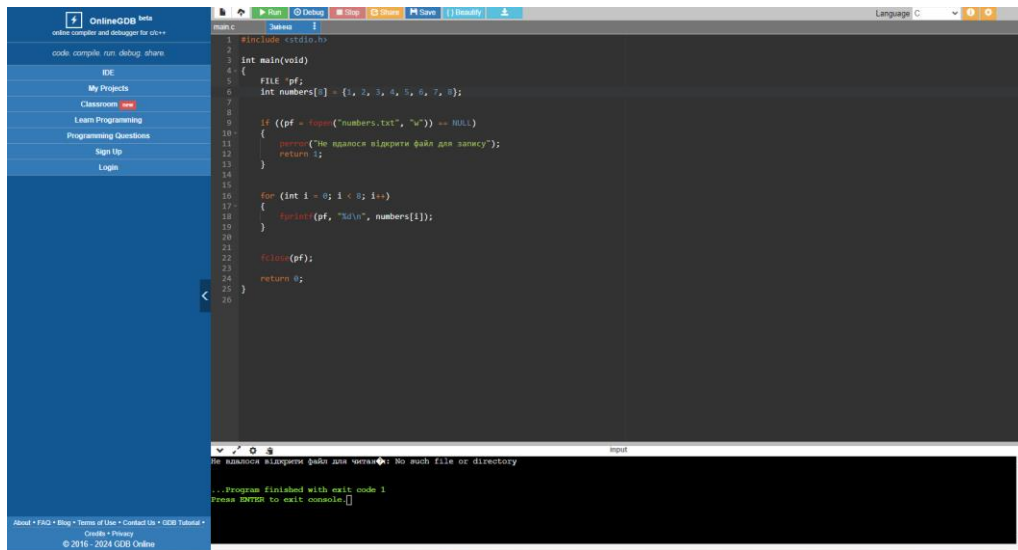
...Program finished with exit code 0
Press ENTER to exit console.



```
1 #include <stdio.h>
2
3 void main(void)
4 {
5     FILE *pf;
6
7     if ((pf = fopen("proba.txt", "w")) == NULL)
8     {
9         perror("proba.txt");
10        return;
11    }
12
13    int k;
14    for (k = 0; k <= 5; k++)
15    {
16        fprintf(pf, "%d %d\n", k, k * k * k);
17    }
18    fclose(pf);
19
20
21
22
23
```

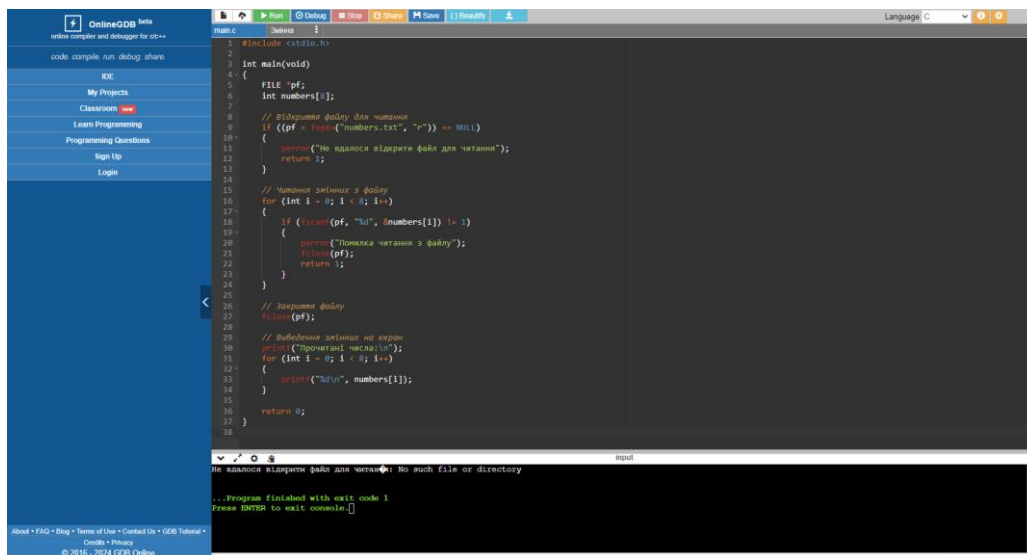
...Program finished with exit code 0
Press ENTER to exit console.

4. Варіант 5



```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     FILE *pf;
6     int numbers[10] = {1, 2, 3, 4, 5, 6, 7, 8};
7
8     if ((pf = fopen("numbers.txt", "r")) == NULL)
9     {
10         perror("Не вдалося відкрити файл для запису");
11         return 1;
12     }
13
14     for (int i = 0; i < 10; i++)
15     {
16         fprintf(pf, "%d\n", numbers[i]);
17     }
18
19     fclose(pf);
20
21     return 0;
22 }
```

...Program finished with exit code 1
Press ENTER to exit console.



```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     FILE *pf;
6     int numbers[10];
7
8     // Відкриття файлу для читання
9     if ((pf = fopen("numbers.txt", "r")) == NULL)
10     {
11         perror("Не вдалося відкрити файл для читання");
12         return 1;
13     }
14
15     // Читання змісту з файлу
16     for (int i = 0; i < 10; i++)
17     {
18         if (!fscanf(pf, "%d", &numbers[i]) != 1)
19         {
20             perror("Помилка читання з файлу");
21             fclose(pf);
22             return 1;
23         }
24     }
25
26     // Закриття файлу
27     fclose(pf);
28
29     // Виведення змісту на екран
30     printf("Прочитані числа:\n");
31     for (int i = 0; i < 10; i++)
32     {
33         printf("%d\n", numbers[i]);
34     }
35
36     return 0;
37 }
```

...Program finished with exit code 1
Press ENTER to exit console.

Контрольні запитання:

1. Суть поняття файлу в мові програмування C

Це абстракція для зберігання даних. Можна сказати що це послідовність байтів, яка зберігається на носії інформації. Для роботи з файлами у C використовується бібліотека `stdio.h`, яка надає засоби для відкриття, читання, запису та закриття файлів. У C файли представлені об'єктами типу `FILE`, які використовуються для управління файлами і забезпечення зручного інтерфейсу для операцій вводу-виводу.

2. Основні режиму відкриття файлу C

Основні режими відкриття файлу в C

В C файли можна відкривати в різних режимах, які визначають, як саме файл буде використовуватись.

Ось основні режими:

"r": Відкриття файлу для читання. Файл повинен існувати.

"w": Відкриття файлу для запису. Якщо файл існує, його вміст буде видалено. Якщо файл не існує, він буде створений.

"a": Відкриття файлу для додавання. Дані будуть записані в кінець файлу. Якщо файл не існує, він буде створений.

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

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

"a+": Відкриття файлу для читання та додавання. Дані будуть записані в кінець файлу. Якщо файл не існує, він буде створений.

3. Основні функції при роботі з файлами в мові C

А) `fopen()`: Відкриває файл і повертає вказівник на файл.

Б) `fclose()`: Закриває файл.

В) `fscanf()`: Читає форматовані дані з файлу.

Г) `fprintf()`: Записує форматовані дані у файл.

Д) `fgets()`: Читає рядок із файлу.

Е) `fputs()`: Записує рядок у файл.

Є) `fread()`: Читає дані з файлу у масив.

Ж) `fwrite()`: Записує дані з масиву у файл.

З) `feof()`: Перевіряє, чи досягнуто кінця файлу.

И) `ferror()`: Перевіряє наявність помилок у файлі.

4. Способи позиціювання в файлі на мові C

А) `fseek()`: Встановлює нову позицію вказівника файлу.

Б) `ftell()`: Повертає поточну позицію вказівника у файлі.

В) `rewind()`: Встановлює вказівник на початок файлу.

Г) `fgetpos()`: Зберігає поточну позицію вказівника файлу.

Д) `fsetpos()`: Відновлює позицію вказівника файлу до збереженої позиції.