

Функції для роботи з файлами



Лекція №15

Дисципліна «Програмування»



Поняття файлу

Файл – це впорядкована сукупність даних, що зазвичай розташована на жорсткому диску або іншому зовнішньому носії інформації.

Великий за розміром файл може зберігатися в декількох окремих фрагментах або містити додаткові дані, які дозволяють операційній системі визначати вид цього файлу.

В мові **с** файл розглядається як неперервна послідовність байтів, кожен з яких може бути прочитаний індивідуально.

Існують два режими представлення файлів: **текстовий** і **двійковий**.



Текстовий і двійковий режими

Вміст усіх файлів зберігається у двійковій формі.

Якщо в файлі двійкові коди символів використовуються для представлення тексту, то такий файл є текстовим.

Якщо ж двійкові значення в файлі представляють код на машинній мові, числові дані (з застосуванням внутрішнього представлення значень, наприклад, **long** або **double**), кодування зображення або музичного твору, то вміст буде двійковим.

Мова **c** надає два способи доступу до файлів: **двійковий режим** і **текстовий режим**.

У двійковому режимі програма має доступ до кожного байту. В текстовому режимі те, що бачить програма, може відрізнятися від того, що зберігається у файлі.



Рівні вводу-виводу

В більшості випадків можна обрати один з двох рівнів вводу-виводу (тобто один з двох рівнів керування доступом до файлів).

Низькорівневий ввід-вивід передбачає використання основних служб вводу-виводу, що надаються операційною системою.

Стандартний високорівневий ввід-вивід передбачає застосування стандартного пакета бібліотечних функцій **C** та визначень з файлу заголовку **stdio.h**.

Стандарт **C** підтримує тільки стандартний пакет вводу-виводу, оскільки немає жодної можливості гарантувати, що усі операційні системи можуть бути представлені однаковою низькорівневою моделлю вводу-виводу.



Стандартні файли

Програми на **c** автоматично відкривають три файли, які називаються стандартним вводом (**stdin**), стандартним виводом (**stdout**) і стандартним виводом помилок (**stderr**).

За замовчуванням стандартний ввід являє собою, як правило, клавіатуру. Стандартний вивід і стандартний вивід помилок за замовчуванням є екраном монітору.

Стандартний ввід забезпечує ввід даних до програми за допомогою функцій **getchar()** і **scanf()**.

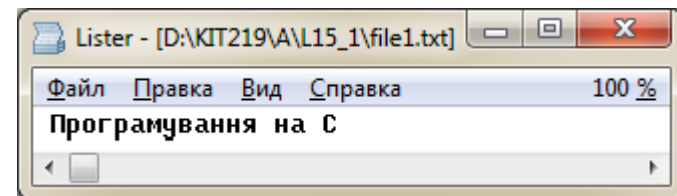
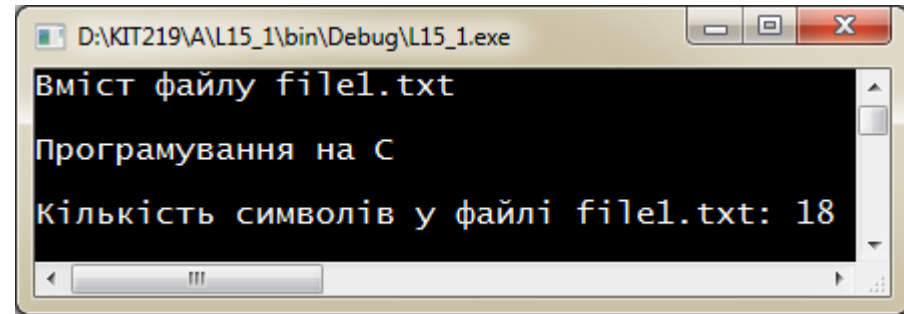
Стандартний вивід забезпечується функціями **putchar()**, **puts()** і **printf()**.

Стандартний вивід помилок надає логічно відокремлене місце для відправлення повідомлень про помилки.

Застосування функцій

```
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
int main(void)
{
    int ch;
    FILE *fp;           // вказівник на файл
    unsigned long count = 0;
    char *name_file = "file1.txt";
    char words[] = "Програмування на C";

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    if((fp = fopen(name_file, "w")) == NULL)
    {
        printf("Неможливо відкрити файл file1.txt\n");
        exit(EXIT_FAILURE);
    }
    else fputs(words, fp);
    fclose(fp);
}
```





Застосування функцій

```
if((fp = fopen(name_file, "r")) == NULL)
{
    printf("Неможливо відкрити файл file1.txt\n");
    exit(EXIT_FAILURE);
}
else
{
    puts("Вміст файлу file1.txt\n");
    while((ch = getc(fp)) != EOF)
    {
        putc(ch, stdout); // те ж саме, що й putchar(ch);
        count++;
    }
    printf("\n\nКількість символів у файлі %s: %lu\n",
        name_file, count);
}
fclose(fp);

return 0;
}
```



Функція `fopen()`

Прототип функції `fopen()` має наступний вигляд:

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

Функція `fopen()` призначена для відкриття файлу.

Вона оголошена в файлі заголовку `stdio.h`.

Її першим аргументом є ім'я файлу, який необхідно відкрити (точніше, це адреса рядка, який містить ім'я файлу). Другий аргумент – рядок, що ідентифікує режим, в якому файл повинен бути відкритий.

Після успішного відкриття файлу функція `fopen()` повертає вказівник файлу, який має тип вказівника на `FILE`, де `FILE` – похідний тип, що визначений у файлі `stdio.h`.



Функція `fopen()`

Вказівник вказує на об'єкт даних, який містить інформацію про файл, включаючи відомості про буфер, який застосовується для файлового вводу-виводу.

Функція `fopen()` повертає нульовий вказівник, якщо їй не вдається відкрити файл.

Коли вказівник дорівнює значенню `NULL`, програма завершує своє виконання.

Функція `fopen()` може відмовити у відкритті файлу через переповнення диску, відсутність файлу в заданому каталозі, неприпустиме ім'я, наявність україномовного або російськомовного шляху до файлу, обмеження доступу або апаратної проблеми.



Режими відкриття файлу

Рядок режиму	Опис
"r"	Відкрити текстовий файл для читання
"w"	Відкрити текстовий файл для запису з усиченням існуючого файлу до нульової довжини або створенням файлу, якщо він не існує
"a"	Відкрити текстовий файл для запису з додаванням даних у кінець існуючого файлу або створенням файлу, якщо він не існує
"r+"	Відкрити текстовий файл для оновлення (читання та запису)
"w+"	Відкрити текстовий файл для оновлення (читання та запису), попередньо зробивши усичення файлу до нульової довжини, якщо він існує, або, створивши файл, якщо його немає



Режими відкриття файлу

Рядок режиму	Опис
"a+"	Відкрити текстовий файл для оновлення з додаванням даних в кінець існуючого файлу або створенням файлу, якщо він не існує; читати можна весь файл, але записувати допускається тільки в кінець файлу
"rb", "wb", "ab", "ab+", "a+b", "wb+", "w+b", "ab+", "a+b"	Подібні до попередніх режимів, за виключенням того, що замість текстового режиму вони використовують двійковий режим
"wx", "wbx", "w+x", "wb+x" або "w+bx"	(C11) Подібні до режимів без літери x , за виключенням того, що вони відмовляються працювати, якщо файл існує, та відкривають файл в монопольному режимі, якщо це можливо



Функції `getc()` і `putc()`

```
int getc(FILE *stream);
```

```
int putc(int ch, FILE *stream);
```

Робота функцій `getc()` і `putc()` схожа з роботою функцій `getchar()` і `putchar()`. Відмінність полягає в тому, що треба вказати, з яким файлом їм необхідно працювати.

```
ch = getchar();           // означає «отримати символ  
                           // зі стандартного вводу»
```

```
ch = getc(fp);            // означає «отримати символ з  
                           // файлу, на інформацію про який  
                           // вказує fp»
```

```
putc(ch, fpout);          // означає «помістити символ ch  
                           // у файл, на інформацію про який  
                           // вказує fpout»
```



Кінець файлу

Програма, яка читає дані з файлу, повинна зупинитися, коли вона досягне кінця файлу. Для цього під час файлового вводу повинен застосовуватися цикл з вхідною умовою.

```
int ch;
FILE *fp;

fp = fopen("input.txt", "r");
while((ch = getc(fp)) != EOF)
{
    putchar(ch);    // вивести поточний символ
}
fclose(fp);
```

Застосування функцій

```
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <string.h>
```

```
int main(void)
```

```
{
```

```
    FILE *in, *out;    // оголошення двох вказівників на FILE
```

```
    int ch;
```

```
    char *name_in  = "input.txt";
```

```
    char *name_out = "output.txt";
```

```
    int count = 0;
```

```
    SetConsoleCP(1251);
```

```
    SetConsoleOutputCP(1251);
```

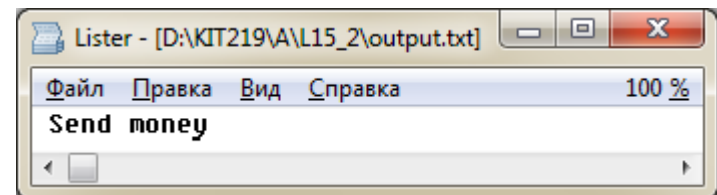
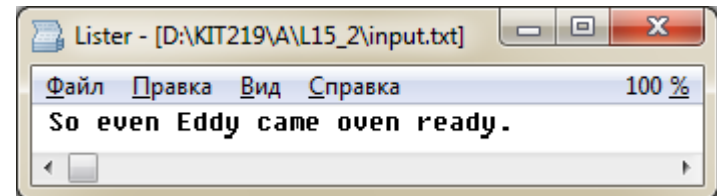
```
    if((in = fopen(name_in, "r")) == NULL)
```

```
    {
```

```
        fprintf(stderr, "Неможливо відкрити файл input.txt\n");
```

```
        exit(EXIT_FAILURE);
```

```
    }
```





Застосування функцій

```
else
{
    if((out = fopen(name_out, "w")) == NULL)
    {
        printf("Неможливо відкрити файл output.txt\n");
        exit(EXIT_FAILURE);
    }
    else
    {
        while((ch = getc(in)) != EOF)
        {
            if(count % 3 == 0)
                putc(ch, out);
            count++;
        }
    }
}
if(fclose(in) != 0 || fclose(out) != 0)
    fprintf(stderr, "Помилка під час закриття файлів\n");
return 0;
}
```

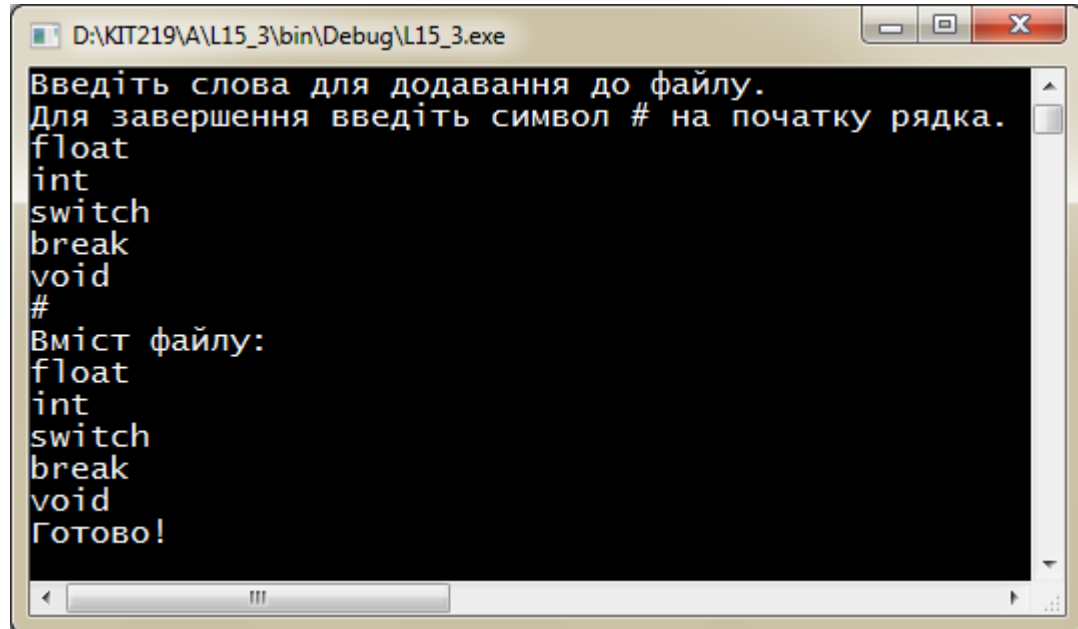
Застосування функцій

```
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#define MAX 41

int main(void)
{
    FILE *fp;
    char words[MAX];

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    if((fp = fopen("input.txt", "a+")) == NULL)
    {
        fprintf(stdout, "Не вдається відкрити файл"
                  " input.txt\n");
        exit(EXIT_FAILURE);
    }
}
```



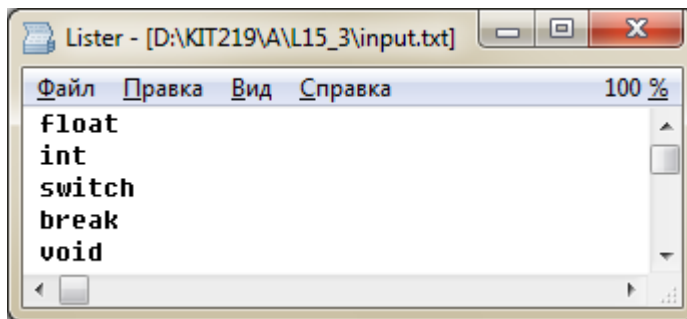
```
D:\KIT219\A\L15_3\bin\Debug\L15_3.exe
Введіть слова для додавання до файлу.
Для завершення введіть символ # на початку рядка.
float
int
switch
break
void
#
Вміст файлу:
float
int
switch
break
void
Готово!
```


Застосування функцій

```
puts("Введіть слова для додавання до файлу.");
puts("Для завершення введіть символ # на початку рядка.");
while((fscanf(stdin,"%40s", words) == 1) &&
      (words[0] != '#'))
    fprintf(fp, "%s\n", words);
puts("Вміст файлу:");

rewind(fp);           // повернення на початок файлу

while(fscanf(fp,"%s",words) == 1)
    puts(words);
puts("Готово!");
if(fclose(fp) != 0)
    fprintf(stderr, "Помилка під час закриття файлу\n");
return 0;
}
```





Функція fseek()

```
int fseek(FILE *stream, long offset, int origin);
```

stream – вказівник типу **FILE** на файл, в якому буде відбуватися пошук;

offset – зміщення – показує, наскільки далеко необхідно переміститися від стартової точки. В цьому аргументі необхідно передавати значення типу **long**, яке може бути **додатним** (переміститися вперед), **від'ємним** (переміститися назад) або **нульовим** (залишитися на місці);

origin – встановлює режим, що ідентифікує стартову точку.

Режим	Звідки вимірюється зміщення	Значення
SEEK_SET	Від початку файлу	0
SEEK_CUR	Від поточної позиції	1
SEEK_END	Від кінця файлу	2



Функції `fseek()` і `ftell()`

Функція `fseek()` дозволяє трактувати файл подібно масиву і переходити безпосередньо до будь-якого байту в файлі, що відкритий за допомогою `fopen()`.

```
// перейти на початок файлу
fseek(fp, 0L, SEEK_SET);
// перейти вперед на 10 байтів від початку
fseek(fp, 10L, SEEK_SET);
// перейти вперед на 2 байти від поточної позиції
fseek(fp, 2L, SEEK_CUR);
// перейти в кінець файлу
fseek(fp, 0L, SEEK_END);
// перейти назад на 10 байтів від кінця файлу
fseek(fp, -10L, SEEK_END);
```

Функція `ftell()` має тип `long` і повертає поточну позицію у файлі.

```
long ftell(FILE *stream);
```

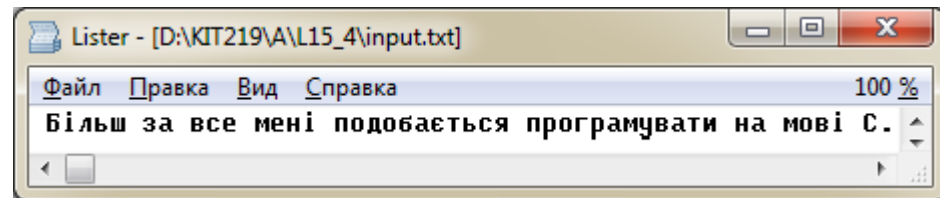
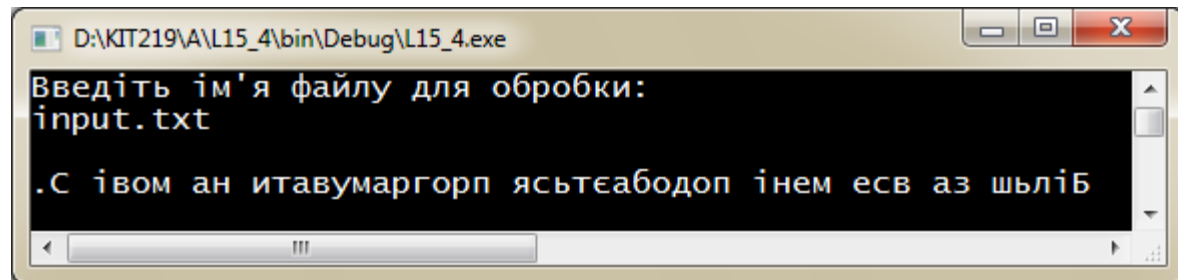
Застосування fseek() і ftell()

```
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#define CNTL_Z '\032' // маркер кінця файлу
#define SLEN 81
```

```
int main(void)
{
    char file[SLEN];
    char ch;
    FILE *fp;
    long count, last;

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    puts("Введіть ім'я файлу для обробки:");
    scanf("%80s", file);
```





Застосування fseek() і ftell()

```
if((fp = fopen(file, "rb")) == NULL)
{
    // режим тільки для читання
    printf("Файл неможливо відкрити %s\n", file);
    exit(EXIT_FAILURE);
}
fseek(fp, 0L, SEEK_END); // перейти в кінець файлу
last = ftell(fp);
for(count = 1L; count <= last; count++)
{
    // рухатися у зворотному напрямку
    fseek(fp, -count, SEEK_END);
    ch = getc(fp);
    if(ch != CNTL_Z && ch != '\r') // файли MS DOS
        putchar(ch);
}
putchar('\n');
fclose(fp);
return 0;
}
```



Функції `fgetpos()` і `fsetpos()`

```
int fgetpos(FILE *restrict stream, fpos_t *restrict pos);
```

Виклик `fgetpos()` поміщає поточне значення типу `fpos_t` в комірку, яка вказується вказівником `pos`. Це значення описує позицію в файлі. Функція повертає `0` у випадку успіху та ненульове значення у випадку відмови.

```
int fsetpos(FILE *stream, const fpos_t *pos);
```

Виклик `fsetpos()` призводить до використання значення типу `fpos_t` з комірки, яка задається за допомогою `pos`, для встановлення вказівника файлу в позицію, що містить це значення. Функція повертає `0` у випадку успіху та ненульове значення у випадку відмови.

Значення `fpos_t` повинне бути отримане попереднім викликом функції `fgetpos()`.



Функція `ungetc()`

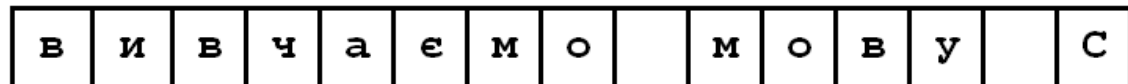
```
int ungetc(int c, FILE *fp);
```

Функція `ungetc()` повертає символ, що вказаний в `c`, назад до вхідного потоку. У випадку повернення символу до вхідного потоку він буде прочитаний наступним викликом стандартної функції вводу

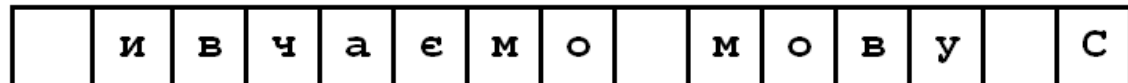
Оператор

Вхідна черга символів

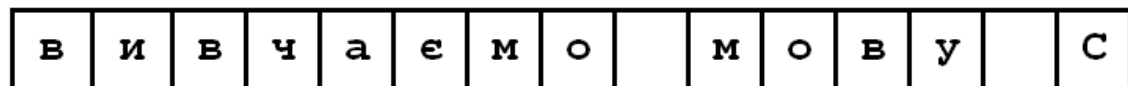
(Початковий стан)



```
ch = getchar();
```



```
ungetc(ch, stdin);
```





Функція fflush()

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

Виклик функції `fflush()` призводить до того, що будь-які дані в буфері виводу, які ще не були записані, відправляються у вихідний файл, що ідентифікується за допомогою `fp`. Цей процес називається **скиданням буфера**.

Якщо `fp` – нульовий вказівник, то скидаються усі буфери виводу. Результат використання функції `fflush()` на вхідному потоці є невизначеним.

Функцію `fflush()` можна застосовувати з потоком оновлення (для будь-якого режиму читання-запису), за умови, що найостанніша операція, яка використовувала потік, не була операцією вводу.



Функція `setvbuf()`

```
int setvbuf(FILE *restrict fp, char *restrict buf,  
            int mode, size_t size);
```

Функція `setvbuf()` встановлює альтернативний буфер, що призначений для застосування стандартними функціями вводу-виводу. Вона викликається після того, як файл був відкритий, і перед виконанням будь-якої іншої операції з потоком даних. Вказівник `fp` ідентифікує потік, а `buf` вказує на місце зберігання. Аргумент `size` повідомляє `setvbuf()` розмір цього масиву.

Для `mode` доступні наступні варіанти:

`_IOFBF` означає повну буферизацію (буфер скидається, коли він повний);

`_IOLBF` означає порядкову буферизацію (буфер скидається, коли він повний або коли в нього записаний символ нового рядка);

`_IONBF` означає відсутність буферизації.

Функція `setvbuf()` повертає `0` у разі успіху і ненульове значення в іншому випадку.