

# **Функції потокового введення/виведення даних**

Виділяють чотири групи функцій високорівневого обміну відповідно до формату даних, що передаються

за одну операцію:

- посимвольного введення/виведення;
- рядкового введення/виведення;
- блокового введення/виведення;
- форматного введення/виведення.

Всі операції виконуються , починаючи з поточної позиції

файла, яку зберігає спеціальний покажчик. У процесі введення/виведення покажчик автоматично

# Посимвольний обмін даними

Введення одного символу з потоку *fp*

***int fgetc (FILE \*fp);***

Функція повертає код зчитаного символу. Якщо ж читання з потоку недоступне, повертається макроконстанта EOF. Такі ж дії виконує функція

***int getc (FILE \*fp)***, яку в Borland C оголошено як макрос.

Функція ***int ungetc (int symb, FILE \*fp);***

дає змогу повернути в потік введення *fp* останній зчитаний символ (після повернення *symb* стає першим символом потоку).

*Приклад: зчитування послідовності цифрових символів з потоку frd та запис цифр за адресою numbet*

```
#include <stdio.h>  
#include <ctype.h>  
char * ReadNum (char *numbst, FILE * frd)  
{  
    char * pn = numbst;  
    int symb;  
    while (isdigit(symb = getc(frd)))    /* виявлення  
                                           десяткової цифри */  
        *pn++ = symb;    /* запис цифрових символів у  
numbst* |  
    *pn = `\\0`;    /* кінець числового рядка */  
    ungetc (symb, frd);    /* повернення нечислового символу */  
    return numbst;  
}
```

Запис одного символу *symb* у потік виведення *fp*  
виконують функції

***int fputc (int symb, FILE \*fp);***

***int putc(int symb, FILE \*fp);*** - макрос Borland C

За умови успішного виконання функції  
повертають

код записаного символу, у разі помилки  
звертання

до файлу – константу EOF.

# Файловий обмін рядками символів

Зчитування рядка символів з потоку

***char \*fgets (char \*str, int max, FILE \*fp);***

*str* – масив символів, в який буде записано введений рядок; *max* – максимальна кількість символів (з нуль-символом включно), яку може містити зчитаний рядок; *fp* – потік, з якого вводяться рядки.

З потоку у ділянку оперативної пам'яті ( *str* ) зчитується

послідовність символів до символа нового рядка чи символа кінця файла, але не більше, ніж *max-1* символів.

У разі успішного виконання функція повертає

Запис заданого рядка в потік виведення

***int fputs (char \*str, FILE \*fp);***

Повертає ненульове значення за умови успішного виконання та EOF у разі невдачі. Функція послідовно передає у потік символи рядка *str* до ``\0``.

*Додатково:* при створенні потоку функцією ***fopen()*** в параметрі *fmode* можна задати текстовий *t* чи бінарний *b* режим відкриття потоку. За замовчуванням встановлюється текстовий режим.

Різниця між двома режимами – в інтерпретації коду клавіші *Enter*. В текстовому режимі цей код в процесі читання замінюється на символом нового рядка ‘*\n*’, а в разі запису навпаки – кожен символ ‘*\n*’ заноситься у потік як комбінація “*\r\n*”. В бінарному режимі такі заміни не виконуються.

## ***Форматне введення/виведення даних***

Файлове введення даних згідно зі заданим СПИСКОМ

Параметр *fp* вказує на текстовий потік введення, обов'язковий параметр *format* – задає символний рядок з послідовністю специфікацій форматних перетворень. Наступні параметри задають адреси змінних, куди будуть записуватись введені значення

(їх кількість і типи визначаються специфікаціями) .

Правила форматних перетворень даних такі ж, як і

для функції *scanf()*.

Помилки в процесі введення можна конкретизувати

через функцію *feof()* – повертає нуль, якщо не встановлено ознаку кінця файлу, інакше повертає



Приклад неправильної організації форматного введення даних

```
FILE *fin;
```

```
int numb;
```

```
.....
```

```
while ( !feof ( fin)) {  
    fscanf ( fin, "%d", &numb);  
    printf ("%3d", numb);  
}
```

Якщо вміст файла є таким:      1 2 3 4 5

де символ підкреслення позначає роздільник, то  
на

екран буде виведено:            1 2 3 4 5

Якщо ж у файлі в кінці буде записано роздільник:

```
1_2_3_4_5_
```

то результат читання та виведення буде іншим:

```
1 2 3 4 5 5
```

Повторення зумовлене тим, що після введення числа 5

не встановлюється ознака кінця файла (є кінцевий незчитаний роздільний символ). Наступний виклик *fscanf()* зчитує цей символ, але не знаходить більше чисел, тому фіксується кінець файла, а значення *numb*

не змінюється. Щоб зробити форматне введення незалежним від прикінцевих символів файла, цикл читання слід записати так:

.....

```
while ( fscanf ( fin, "%d", &numb)==1)
    printf ( "%3d", numb);
if ( feof (fin))
    puts (" Зчитано всі числа файла ");
else
    puts (" Помилка в числових даних ");
```

.....

Для читання символічних даних використовують спеціфікатор "%s", при цьому зчитується послідовність символів до першого роздільника.

В разі звертання до файла *fscanf (f, “%s”, str);* у рядок *str* буде зчитане тільки одне поточне слово. Необхідно забезпечити щоб обсяг ділянки *str* був достатнім для запису найдовшого слова файлу.

Форматне виведення в файл здійснюється функцією

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

Приклад:

```
void vyvod (FILE *f)  
{   int a,b,c,k;  
    scanf (f, “%d%d%d”, &a, &b, &c);  
    fprintf (f, “a=%d, b=%d, c=%d);  
}
```

# Приклади роботи з текстовими файлами

Відкриття текстового файлу *test.txt* може мати вигляд

```
#include<stdio.h>  
void main()  
{    ...  
    FILE *f;  
    if (( f=fopen("test.txt", "rt"))==NULL)  
    {  
        printf("Файл не вдалося відкрити.\n");  
        return;  
    }    ...  
    fclose(f);  
  
    ...  
}
```

Поток  $f$  зв'язується з файлом "test.txt", який відкривається як текстовий тільки для читання.

Після

закінчення роботи з файлом, його необхідно закрити за

допомогою функції *fclose()*. З текстового файла можна

читати інформацію по рядках, по символах або за форматом. Приклад зчитування рядків:

```
#include<stdio.h>  
#include<string.h>  
void main()  
{  
    char s[80];
```

```
{  
    printf ("There are an error\n");  
    return;  
}  
do  
    {  
        fgets (s, 80, f);  
        printf("%s",s);  
    } while ( !feof(f) );  
    fclose (f);  
}
```

Функція *feof()* перевіряє символ завершення файлу.

Якщо такий символ прочитаний, то *feof()* повертає ненульове значення і цикл завершується.

Приклад форматного обміну:

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

```
void main()
```

```
{ FILE *fi;
```

```
int age;
```

```
fi=fopen("age.txt","r"); /* відкриття для читання */
```

```
fscanf(fi,"%d",&age); /*читання числового значення  
*/
```

```
fclose(fi); /* закриття файла */
```

```
fi=fopen("data.txt", "a"); /* відкриття файла для  
додавання інформації в кінець */
```

```
fprintf (fi, "Age==%d.\n",age); /* запис рядка в файл  
*/
```

```
fclose(fi); /* закриття файла */
```

```
}
```



Створення файла з результатами табулювання  
функції

```
# include <stdio.h>  
double fun (double);  
int main(void)  
{  
    FILE *fout;  
    double x0, xk, x, dx;  
    printf (“\n Межі та крок табулювання: “);  
    scanf (“%1f%1f%1f”, &x0, &xk, &dx);  
    fout = fopen( “fun_tab.res”, “wt”);  
    for (x=x0; x<xk; x+=dx)
```

```
fprintf (fout, "%10.21f%15.31f\n", x, fun(x) );  
fclose (fout);  
printf ( "\n Файл результатів створеною \n");  
return 0;  
}  
double fun (double x)  
{  
..... /*тіло функції*/  
}
```