# Обмін блоками даних

Зчитати з потоку блок даних заданого розміру можна за допомогою функції

***size_t fread (void *buf, size_t size, size_t n, FILE *fp);***

що заносить у буфер *n* об'єктів розміром *size.* Тип *size_t*

оголошено в *<stdio.h>* через декларацію *typedef.* У *Borland C* він збігається з типом *int.*

Функція повертає кількість реально зчитаних об' єктів.

Розглянемо приклад визначеня середньоарифметичного значення дійсних чисел, бінарні коди яких зберігаються у файлі. Дані зчитуються в буфер блоками по N чисел. Робота завершується коли розмір зчитаного блоку менший

```
#include <stdio.h>
#define fname "sum.xz"
#define N 100
 void main()
 {
  int a[N], n, i;
  float s=0, k=0;
  FILE *f;
   f =fopen ( fname, "rb");
```

```
do
 {
 n=fread (a, sizeof(int), N, f);
  for (i=0; i<n; i++)
    s+=a[i];
  k+=n;
 }
 while (n==N);
  printf ("sr=%4.2f", s/k);
  fclose(f);
}
```

Запис блоку даних у потік виконує функція

***size_t fwrite (void *buf, size_t size, size_t n, FILE *fp);***

Параметри такі ж, як і в функції ***fread().***

Бібліотека *Borland C* додатково включає функції

***int getw (FILE *fp);***

***int putw (int numb, FILE *fp);***

Функції повертають значення зчитаного або записаного
в бінарний файл двобайтового двійкового коду цілого
числа.

*Приклад: знаходження номера заданого числа в даному бінарному файлі*

```
#include <stdio.h>
long Find(FILE *f, int n)
{
 long k=1,N;
do {
   if (N=getw(f)==n) return k;
   k++;
 } while (N!=EOF);
 return 0;
}
void main()
{
  FILE *f=fopen("prog.xz","rb");
  printf("%d",Find(f,5));
  fclose(f);
}
```

# Робота з бінарними файлами

Бінарні файли мають переваги, порівняно з текстовими при зберіганні числових даних. Операції читання і запису з такими файлами виконуються швидше, так як відсутня необхідність форматування (переведення в текстове представлення та навпаки). Двійкові файли зазвичай мають менший розмір, ніж аналогічні текстові файли. В двійкових файлах можна переміщуватися в будь-яку позицію і читати або записувати дані в довільній послідовності, в той час, як в текстових файлах практично завжди виконується послідовна обробка інформації.

```
#include<stdio.h>
#include<conio.h>
struct mystruct
{      int i;        char ch;
 };
int main(void)
{
     FILE *stream;
     struct mystruct s;
     if ((stream = fopen("test.dan", "wb")) == NULL)
     {         fprintf(stderr, "Неможливо відкрити файл\n");
               return 1;
     }
     s.i = 0;            s.ch = 'A';
     fwrite(&s, sizeof(s), 1, stream);
     fclose(stream);
     return 0;
}
```

Для організації читання даних з файлу в довільному порядку використовується "покажчик файла" (курсор), який визначає поточну позицію у файлі. При читанні даних курсор автоматично зміщується на число прочитаних байтів. Отримати поточну позицію курсору файла можна за допомогою функції *ftell*()

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

Встановлюється поточна позиція курсору у файлі за допомогою функції *fseek*():

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

Ця функція задає зміщення на число байтів *offset* від точки відліку, яка визначається параметром *whence*. Цей параметр може приймати значення 0, 1, 2.

Можливі значення параметра *whence* функції *fseek*

| Константа | whence | Точка відліку |
|---|---|---|
| SEEK_SET | 0 | Початок файлу |
| SEEK_CUR | 1 | Поточна позиція |
| SEEK_END | 2 | Кінець файлу |

Якщо задане значення whence=1, то *offset* може приймати як додатне, так і від'ємне значення, тобто зсув вперед або назад.

Функція *rewind*() переміщує курсор на початок файлу

***void rewind (FILE *stream);***

Те ж саме можна зробити за допомогою *fseek*()

***fseek (stream, 0L, SEEK_SET);***

Встановлення покажчика перед 10-м елементом від кінця файла: ***fseek (f, -10*sizeof(double), SEEK_END);***

*Приклад визначення розміру файлу*

```
#include <stdio.h>
long filesize (FILE *stream);
int main (void)
{
    FILE *stream;
    stream = fopen("test.dan ", "wb+");
    printf("Розмір файла test.dan рівний %ld
байт\n",
            filesize(stream));
    fclose(stream);
    return 0;
}
```

```
long filesize (FILE *stream)
{
   long curpos, length;
   curpos = ftell (stream);        /*збереження поточної
                                   позиції файла*/
   fseek (stream, 0L, SEEK_END);  /*перехід у кінець*/
  length = ftell (stream); /*розмір відповідає
зміщенню*/
    fseek (stream, curpos, SEEK_SET);   /*повернення
                          покажчика на попередню позицію*/
    return length;
}
```

*Приклад:* необхідно знайти добуток матриці a [n][m] на вектор b[m]. В результаті отримаємо вектор c[n].

$$C_i = \sum_{j=0}^{3} a_{ij} b_j$$

Значення елементів матриці та вектора вводимо з файлу *ish.dan*. Результати передаємо на екран та в файл *rez.dan*.

```
#include<stdio.h>
#include<stdlib.h>
#define  N    2
#define  M    4
```

```
void main()
    {
float  a[N][M], b[M], c[N];
int  i,  j;
FILE *p1, *p2;
p1=fopen("ish.dan","r");
if(p1==0)
  {   puts("Файл ish.dan  не открылся");
      exit(1);       }
p2=fopen("rez.dan", "w");
 for(i=0; i<N; i++)
   for (j=0; j<M; j++)
    fscanf(p1, "%f",&a[i][j]);
```

```
  for (j=0; j<M; j++)
fscanf(p1, "%f",&b[j]);
 for(i=0; i<N; i++)
   {
     c[i]=0;
      for (j=0; j<M; j++)
         c[i]+=a[i][j]*b[j];
      fprintf(p2, "c[%i]=%f\n", i, c[i]);
     printf("c[%i]=%f\n", i, c[i]);
     }
fclose(p1);
fclose(p2);
  }
```