

24-MA'RUZA. DASTURLASHDA FAYLLAR BILAN ISHLASH. IOSTREAM SINFI

2. Faylga yozish. ofstream sinfi

Yuqorida ko'rib chiqqan va faylga yozish uchun mo'ljallangan ifstream sinfining teskari vazifasini bajaradi.

Ushbu ma'lumotlar turi chiqadigan fayllar oqimini ifodalaydi va fayllarni yaratish va fayllarga ma'lumot yozish uchun ishlatiladi.

ifstream turida bo'lgani kabi, faylni ochish yoki yaratish uchun konstruktor yoki **open()** metodi qo'llaniladi.

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    ofstream file;

    // Fayl yaratish
    file.open("d:\\misolfayl.txt");

    // Ma'lumot qo'shish uchun ochish
    file.open("d:\\misolfayl.txt");
    return 0;
}
```

Quyidagi konstruktor orqali ham bajarish mumkin:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{

    // Fayl yaratish
    ofstream file("d:\\misolfayl.txt");

    // Ma'lumot qo'shish uchun ochish
    ofstream file1("d:\\misolfayl.txt", ios_base::app);
    return 0;
}
```

```
}
```

Fayllar bilan ishlash uchun standart funksiyalar

Agar mavjud faylning oxiriga qo'shishingiz kerak bo'lsa, **ios_base::app** parametri ko'rsatiladi.

Faylning ochiqqligini tekshirish uchun xuddi shu `is_open()` metodi javobgardir.

```
if( file.is_open())
    cout << "Fayl yaratildi" << endl;
else {
    cout << "Fayl topilmadi";
    cin.get();
    return -1;
}
```

Bu tamoyil bir xil. Fayl o'zgaruvchisida mantiqiy ifoda ishlatib, faylning ochiqqligini tekshirish ham mumkin:

```
if ( !fileo ) cout << "Fayl yaratilmadi" << endl;
```

`<< operator.` Formatlangan chiqishni faylga yo'naltiradi. Ushbu printsip `iostream` dagi analog bilan bir xil.

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    ofstream file;
    // Fayl yaratish
    file.open("d:\\misolfayl.txt");

    // Satr yozish
    string s = "Salom dunyo";

    // Haqiqiy son
    double d = 123.456;
```

```
file << s << endl << d << endl;
```

```
return 0;
```

```
}
```

endl operatori. iostream dagi operatorga o'xshab, u matnli fayllarda yangi qatorga o'tishni amalga oshiradi.

write metodi. Ikkilik fayllarda xotira blokini (bayt massivi) faylga qanday bo'lsa shunday yozish uchun foydalaniladi. Har qanday o'zgaruvchi ham baytlar massivi, aniqrog'i, uni shu tarzda ko'rib chiqish mumkin. Shunga ko'ra, ushbu usul o'zining mashina tasvirini faylga yozadi (xotiradagi ko'rinishi).

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <cstring>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    ofstream file("d:\\misolfayl.txt");
```

```
    char *sc = "Matnlar qatori\n";
```

```
    file.write(sc,strlen(sc));
```

```
    int k = 123;
```

```
    file.write((char*)&k,sizeof(k));
```

```
    double dd = 456.789;
```

```
    file.write((char*)&dd,sizeof(dd));
```

```
    return 0;
```

```
}
```

Ushbu metod ikkita parametрни o'z ichiga oladi: ma'lumotlar blokiga ko'rsatkich va ushbu blok egallagan baytlar soni. Masalan, satr strlen() baytni, sizeof() butun sonini egallaydi (bu 32-bitli operatsion tizimda 4 ta, butun son uchun 8 ta va haqiqiy uchun 8 ta).

Shuni yana bir bor ta'kidlash kerakki, << operatori tomonidan formatlangan chiqishdan farqli o'laroq, write() usuli matnli ko'rinishda ma'lumotlarni chiqarmaydi.

close() metodi. close() metodi faylni yopadi. Yozish uchun ochilgan fayllar uchun, o'qish uchun fayllardan farqli o'laroq, faylni yopish majburiydir. Yopiq bo'lmagan fayl ma'lumotlarni qabul qilmasligi mumkin. Ushbu effekt operatsion tizimning o'zi bufferlashtirish tufayli yuzaga kelishi mumkin, chunki faylga

kiritilgan ma'lumotlar aslida xotirada saqlanadi va to'g'ridan-to'g'ri faylga o'tmaydi. Ma'lumotlarni birlashtirish vaqti kelganida operatsion tizim qaror qiladi.

Bunday "kechiktirilgan" bo'shatish "majburiyat" deb nomlanadi (lotincha "commit" so'zidan). Ushbu effekt ma'lumotlar bazasini boshqarish tizimlari tomonidan juda yaxshi qo'llaniladi, bu yerda kiritilgan yozuvlar xotiraga tushadi (tranzaksiya deb ataladi). Faqat maxsus buyruqdan so'ng ular ma'lumotlar bazasi faylining o'ziga ko'chiriladi. **close()** metodi - bu fayl bilan birga tranzaksiyani yopadigan bunday buyruqning misoli hisoblanadi.

Shuni eslatib o'tish joizki, agar siz faylni yopmasdan ma'lumotlarni kiritishingiz kerak bo'lsa, **flush()** metodidan foydalanishingiz kerak.

file.flush();

Yozish uchun qoldirilgan ma'lumotlar faylga o'tadi, ammo yozish uchun hali ham ochiq bo'ladi. Ushbu metod kamdan-kam holatda ishlatiladi, ammo bu haqda bilish foydalidir.

width, precision formatlash metodlari. `iostream` da bo'lgani kabi, `<<` operatori tomonidan chiqariladigan ma'lumotlarni formatlash usullari fayldagi ma'lumotlarni yaxshi belgilash uchun ishlatilishi mumkin.

width() ko'rsatilgan qiymatga mos keladigan belgilar kengligini va **precision()** haqiqiy sonning kasr qismini belgilaydi. Eng oddiy misol, trigonometrik funksiya qiymatlarini jadval orqali matnli faylga chiqarish:

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <math.h>
using namespace std;
int main()
{
    ofstream file("d:\\misolfayl.txt");

    double d = 1; int i;

    for ( i = 0; i < 10; i++ ){
        d += sin(i/d);

        file.width(20);

        file << i;
```


```

        file.width(20);
        file.precision(5);

        file << d;
        file<<endl;
    }
    return 0;
}

```

Matnli faylda hosil bo‘lgan qiymatlar:



0	1
1	1.8415
2	2.7263
3	3.6177
4	4.5114
5	5.4064
6	6.302
7	7.198
8	8.0944
9	8.9909

14-rasm. “misolfayl.txt” faylida yozilgan natija

seekp, tellp joylashtirish usullari. Fayl bo‘ylab harakatlanish uchun, ifstream holatidagi kabi, pozitsiyani almashtirish funksiyasi mavjud. U **seekp()** deb nomlanadi va **seekg()** uchun yuqorida tavsiflangan parametrlarni oladi.

Faylning boshidan baytlarda joriy holatni olish uchun shunga o‘xshash **tellp()** funksiyasidan foydalaning.

